GEM

Gigabit Ethernet ASIC Specification

Rev 1.2

Part #: 100-5234-100

Products Rights Notice:

You understand that these materials were not prepared for public release and you assume all risks in using these materials. These risks include, but are not limited to errors, inaccuracies, incompleteness and the possibility that these materials infringe or misappropriate the intellectual property right of others. You agree to assume all such risks.

THESE MATERIALS ARE PROVIDED BY THE COPYRIGHT HOLDERS AND OTHER CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS (INCLUDING ANY OF OWNER'S PARTNERS, VENDORS AND LICENSORS) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THESE MATERIALS, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Sun, Sun Microsystems, the Sun logo, Solaris, OpenSPARC T1, OpenSPARC T2 and UltraSPARC are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X / Open Company, Ltd. The Adobe logo is a registered trademark of Adobe Systems, Incorporated. Part of the products covered by these materials may be derived from the Berkeley BSD systems licensed by the University of California. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product described in these materials. This distribution may include materials developed by third parties who have intellectual property rights therein. Products covered by and information contained in these materials may be controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists may be prohibited.

Preface

This documents represents the GEM ASIC specification. GEM implements the MAC (Media Access Control), as well as a subset of the PHY (Physical) layer functions for Ethernet in the speed range of 10 Mbps to 1Gbps. GEM is designed to interface hosts over a PCI bus.

Who Should Use This Specification

This specification is mainly intended for the ASIC, board, and software driver developers involved in Gbps Ethernet. It may also be of interest to architects and designers responsible for host PCI interface definition and development, as well as physical layer network interface developers.

Before You Read This Book

This books references a number of other documents from other parties. Readers are assumed to be familiar with the following:

- *PCI Local Bus Specification, Revision 2.1 (802-2387-02)*
- *Reconciliation Sublayer and Media Independent Interface Specification, 802.3u-1995 or later*
- *Fibre Channel FC-PH Revision 4.3*
- *Fibre Channel - 10-bit Interface. Draft proposed X3 Technical Report, Rev 2.2 December 1995*

Organization

The First chapter includes a brief introduction of the architecture, features, and applications of the device.

Chapter 2 covers the details of the hardware blocks, data structures, external interfaces, and the main internal interfaces.

Chapter 3 is the programmers model, including the register maps and recommended programming practices.

Chapter 4 provides the pinout and pin description of the device, as well as packaging and mechanical information.

Contents

Figures

Tables

Introduction 1

1.1 Purpose of Document

The purpose of this document is to describe the PCI Gigabit Ethernet MAC (GEM) ASIC, while documenting its internal architecture, programmer and external hardware interfaces. Reader familiarity with PCI platforms and Local Area Networks is assumed.

1.2 GEM ASIC objectives

The GEM ASIC represents the main building block for PCI network interfaces capable of transporting data at 1Gbps. The product objective is to allow and optimize the attachment of PCI based hosts to networks based on the specifications under development by the IEEE 802.3z project. Standards compliance on the host PCI interface and network interface are primary objectives of GEM. GEM is intended to provide an expeditious implementation by maximizing the commonality with related projects, and deliver increased levels of network performance to PCI hosts.

While Gigabit Full Duplex operation is the most demanding configuration, GEM also implements a Half Duplex mode, and can be used, if desired, at 100Mbps speeds to interface 100BASE-T Half and Full Duplex networks.

1.3 Features

GEM's features can be classified into *System* and *Network* related features. *System* features affect the GEM to host interface. *Network* features are functions that map to the Media Access Control (MAC) layer in the ISO model hierarchy, and in some cases even to the PHYSICAL (PHY) layer.

- System Features
  - 64/32 bit PCI, up to 66MHz speed
  - 3.3V and 5V signaling for universal PCI cards

- PCI 2.1 compliant
- Register slave and DMA Bus Master interfaces
- Multiple packet buffering in built-in FIFOs
- Manages host queues via Descriptor Rings
- 64 bit DMA address space
- H/W assist for TCP checksum generation/checking
- Fcode ROM support
- Serial EEPROM interface (bit banging)

- Network Features
  - 10/100/1000 Mbps speeds
  - Half and Full Duplex modes
  - Frame Based Link Level Flow Control (IEEE 802.3x)
  - 8B/10B coding and link initialization
  - Network interfaces:

    Bit serial 1.25Gbps PECL differential interface (1Gbps effective)

    Parallel interface software configurable as:

    Nibble-wide single ended MII interface for 100 Mbps

    Byte-wide GMII extensions for 1 Gbps

    10-bit interface for use with external 1Gbps SERDES devices

1.4 Applications

Initially, the first beneficiaries of Gigabit Ethernet speeds are likely to be server links and inter-switch links, with other high performance hosts eventually upgrading to Gigabit Ethernet as well. GEM's main application space is for such server and host network interfaces.

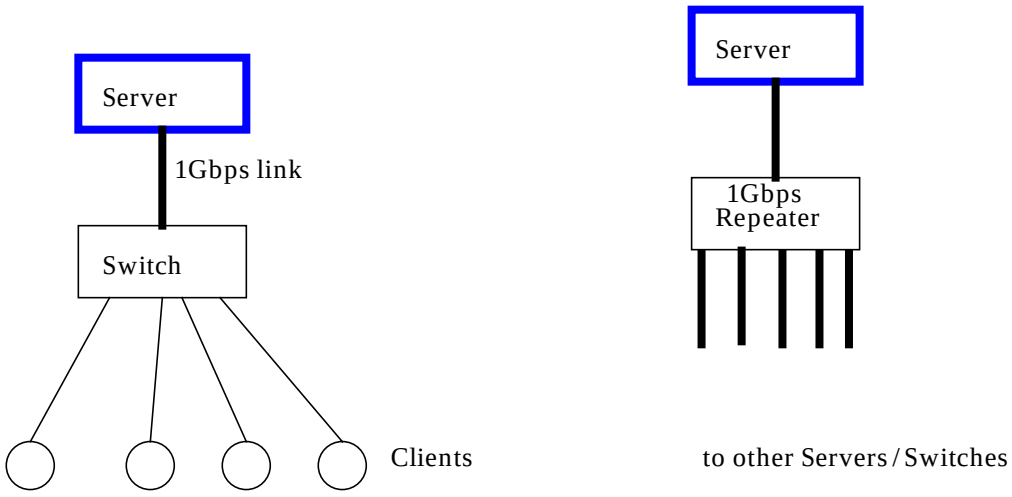


Figure 1-1 Examples of Server interfacing Gigabit links

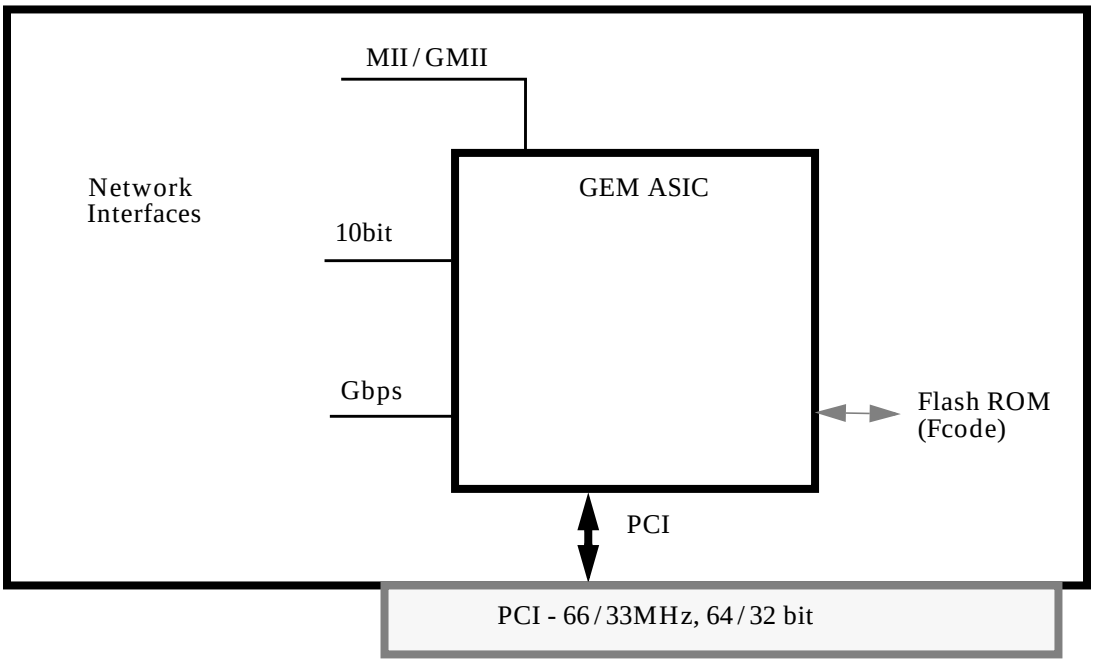
Even though GEM can be used on any PCI host, ultimately the performance achievable will be a function of the PCI bus speed, bus width, the CPU processing power, and the overall host system architecture, as it translates into actual bandwidth and latency between GEM and system memory.

For Sun PCI platforms, where more than one independent PCI bus segment is provided, GEM at 1Gbps bit rates is intended to reside on the highest performance PCI slots (highest speed, widest data path, and possibly dedicated bus slot).

1.5 Basic Architecture

The following basic architecture diagram shows the main GEM interfaces.

Figure 1-2 GEM interfaces



PCI - Universal voltage, up to 66MHz, 32/64 bit interface

MII/GMII - Media Independent Interface: 25MHz nibble wide/125MHz byte wide.

10 bit - 125MHz balanced code interface to external SERDES

Gbps - 1.25Gbps serial differential interface

Flash ROM - 8 bit interface to external Fcode Flash ROM

The first order GEM queue management guidelines can be illustrated using the following conceptual queues shown in Figure 1-3:

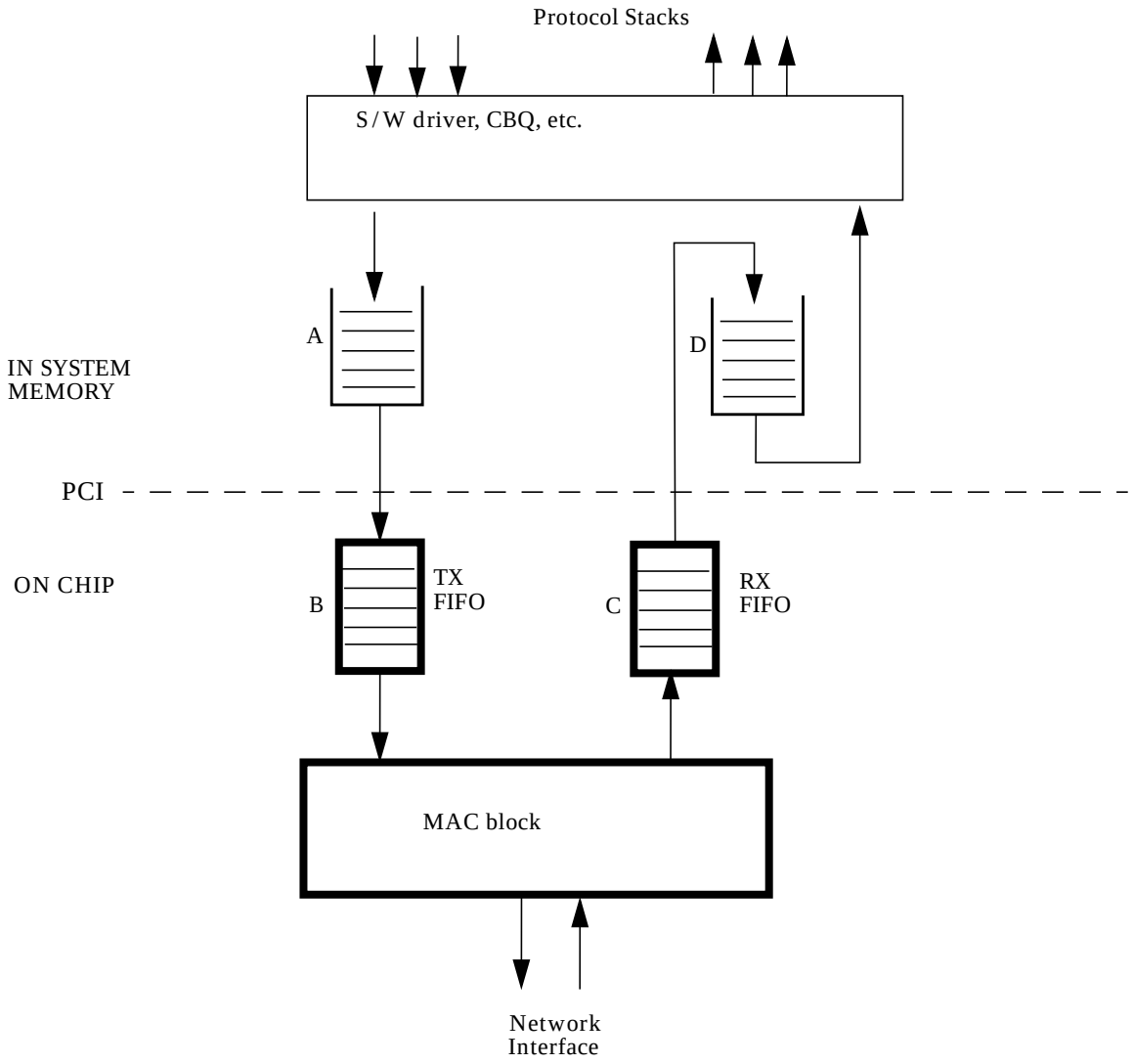


Figure 1-3 Transmit and Receive queues managed by GEM

1. Queues A and D are implemented in system memory using one level of indirection (ring of descriptors with pointers to buffers).
2. Transmit packets in queue A can be fragmented into several buffers.
3. Receive packets in queue D cannot be fragmented into separate buffers.
4. TX FIFO B must be deep enough to prevent under-utilization of the network due to the effects bus latency and receive DMA activity. It should store at least one full packet.
5. RX FIFO C must accommodate the variance in DMA service time and in usable bus bandwidth with acceptably low packet loss probability.
6. Packet flow between queues A to B, and C to D is DMA based.

7. CPU service of queue D is interrupt driven.
8. Transmit packet flow from queue A to B can generate interrupts on specific packets, for scheduling and rate estimation. Interrupts may also be solicited for empty queue conditions for either A, or for A and B.
9. Removal from queue B is normally open ended and cannot generate interrupts. The drain rate might vary as a function of the link speed, collisions, and flow control.
10. If queue C fills up:
    a. It can trigger flow control
    b. If necessary, packets are internally discarded at C's input

GEM's values for the above mentioned queues are:

TX FIFO (B) - 9 kbytes

RXFIFO (C) - 20 kbytes

Descriptor Rings (A and D) - Up to 8k packets per queue

The Functional Description chapter covers GEM queues in more detail.

Functional Description 2

2.1 Overview

GEM incorporates most of the functions required to implement a single-function Gigabit Ethernet PCI network interface. Figure 2-1 shows a top level diagram of GEM's functional blocks. The functions implemented in each block are:

1. PCI Interface
    a. PCI configuration space
    b. Slave Memory mapped I/O registers
    c. DMA bus master interface
    d. Arbitration between DMA channels
2. DMA engines
    a. One DMA channel for transmit and one for receive
    b. Descriptor ring DMA management
    c. 64 bit DMA address space
3. Transmit and Receive FIFOs
    a. Absorb bus access latencies and bus bandwidth variance
    b. De-couple of bus bandwidth from network functionality
    c. Generate Transmit TCP checksum
    d. Lossless flow controlled reception
    e. Prevent re-transmission and bad received frames from consuming system bus bandwidth
4. MAC
    a. Framing and data delineation
    b. CRC checking and generation

c. Unicast and Multicast Address Filtering

d. Flow Control MAC Frame reception and enforcement

e. Flow Control MAC Frame transmission

f. CSMA/CD protocol in Half Duplex

g. Independent Transmit and Receive functions in Full Duplex mode

5. PCS (Physical Coding Sublayer)

a. Encodes transmit data into 10-bit codes

b. Receive synchronization using 8B/10B sync character

c. Decodes 10-bit receive data codes

d. Out of band signaling using special codes

e. Code error detection

f. Link Initialization

6. Serial Link

a. Serialize/Deserialize balanced 10-bit codes

b. Receive clock recovery

c. Differential serial interface to driver optics

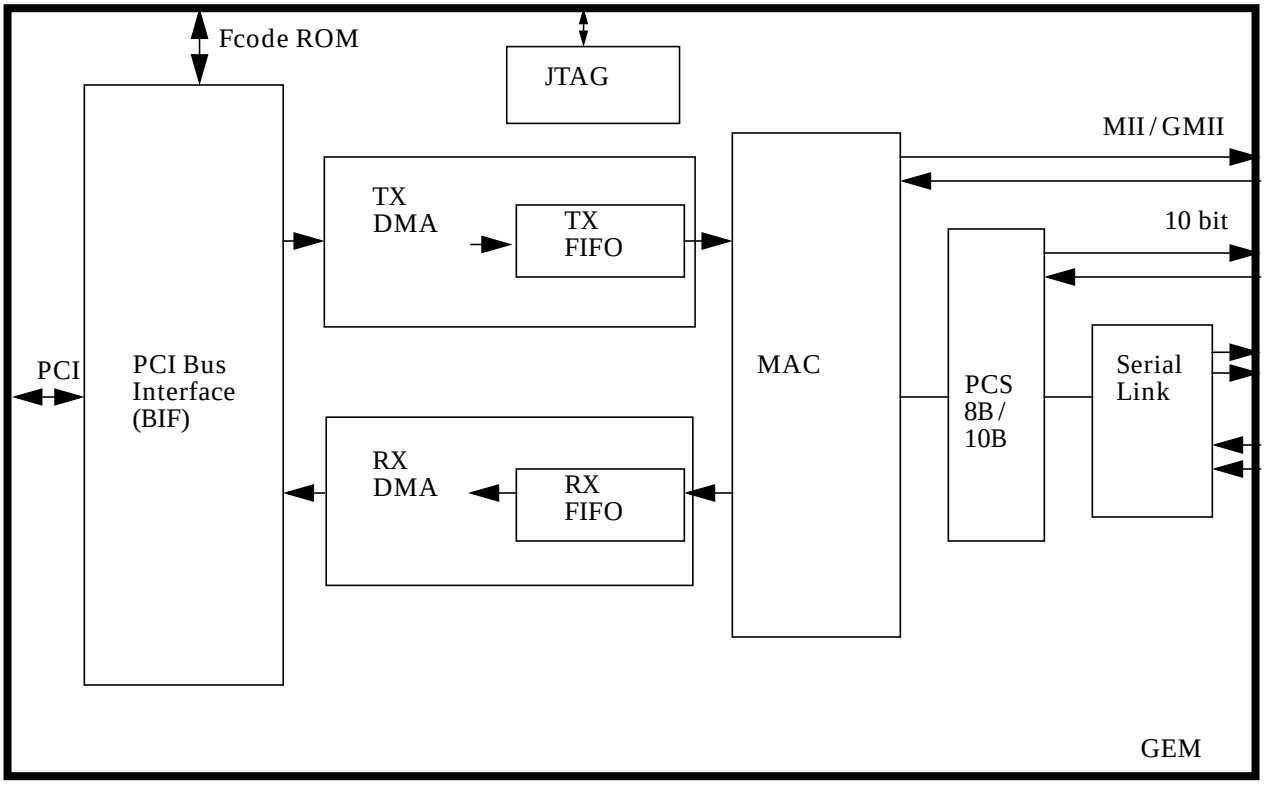


Figure 2-1 GEM Functional Blocks

2.2 Hardware Architecture

2.2.1 Terminology/Conventions

double-word - Used in this specification when refering to 64-bit fields in data structures or internal GEM interfaces

frame - Stream of bits that flows between two nodes at the MAC layer. Includes the Preamble, the SFD, the Destination and Source Addresses , the Length/Type field, the Data/Pad bytes, and the FCS. All frame octets, except the FCS octets, follow a low order first format (least significant bit is transmitted first). The Length/Type field is transmitted with the high order octet first.

MAC address - 48 bit addresses respresented in this specification as 6 bytes of the form *a:b:c:d:e:f*. For globally administered addresses *abc* comprises the Organization Unique Identifier (OUI) part of the address. The least significant bit of *a* is the Multicast bit, and appears first in the media.

2.2.2 Clock Domains

GEM's clock domains are illustrated in Figure 2-2.

2.2.2.1 System Clock Domain

This clock is sourced by the system bus and is defined to operate in the range of up to 66.67MHz. It is used to drive the BIF, the FIFOs and the DMA block.

2.2.2.2 TxMAC Clock Domain

This clock is used to drive the Transmit Protocol Engine in the TxMAC. In MII mode it is sourced by an MII external transceiver and is defined to operate at 2.5/25MHz +/-100ppm. For 1Gbps operation the TxMAC clock is a 125MHz byte clock. This byte clock is either the 125MHz TBC byte clock of the SERDES (internal or external) function, or the GMII transmit clock. The synchronization between this clock domain and the Local clock domain is performed across the Synchronization Fifo in the TxMAC.

2.2.2.3 RxMAC Clock Domain.

This clock is used to drive the Receive Protocol Engine in the RxMAC. In MII mode it is sourced by an external MII transceiver and is defined to operate at 2.5/25MHz +/-100ppm. For 1Gbps operation the RxMAC clock is a 125MHz byte clock. This receive byte clock is directly supplied to the RxMAC either by the SERDES (internal or external) function, or by the GMII interface. For external SERDES devices connected to the 8B/10B interface, the RxMAC clock is derived from the rising edges of the RBC[0] RBC[1] inputs. The synchronization between the RxMAC clock domain and the Local clock domain is performed across the Synchronization Fifo in RxMAC.

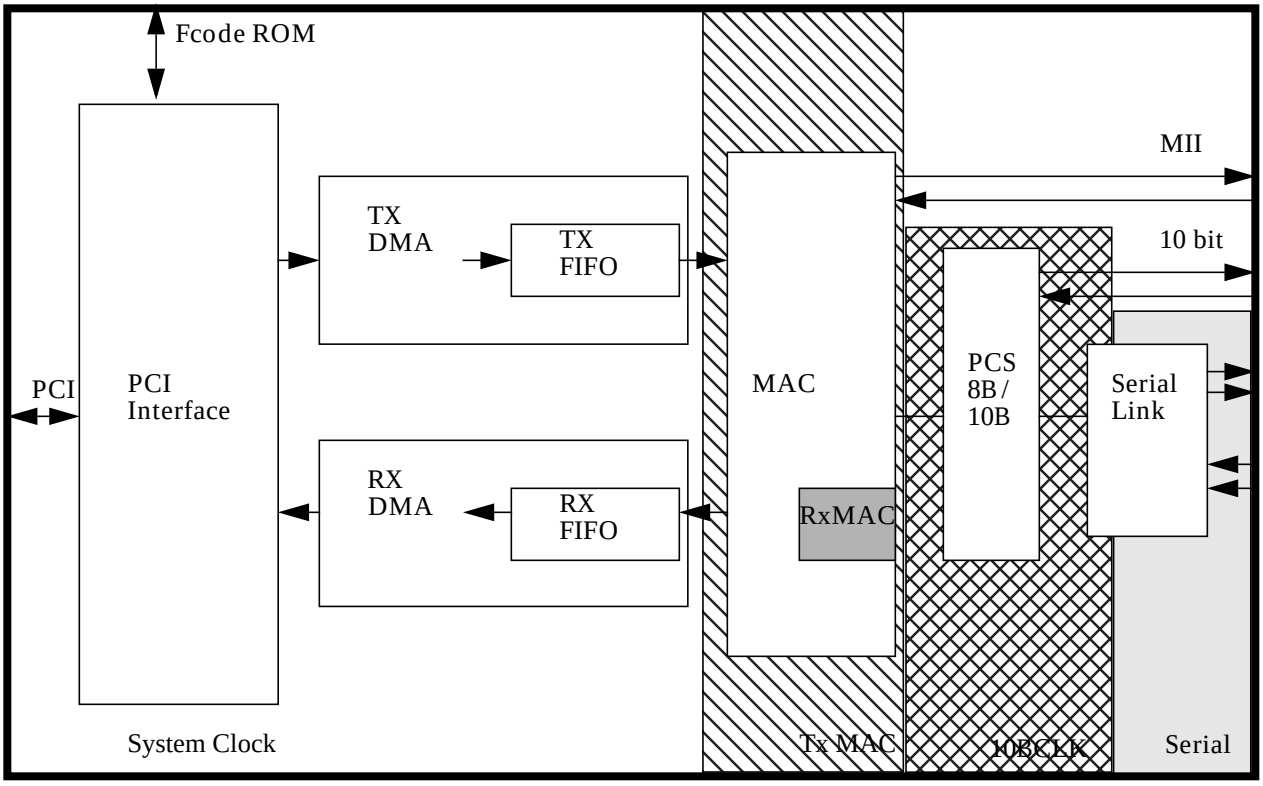


Figure 2-2 Clock Domains

2.2.2.4 *Serial Clock Domain*

The Serial Link block relies on the 125MHz REFCLK input, and it achieves the 1.25Gbps resolution by internally generating phase delayed versions of the 125MHz clock.

2.2.3 *Internal Data Paths*

Internally there are two logically independent data paths, transmit and receive. These two paths meet in the PCI Block, as both paths must share a single PCI bus interface. The paths also meet in the MAC block, where transmit and receive interact in half duplex mode.

While the data throughput over the PCI and the network might be variable, the internal data paths between these boundaries are designed to sustain the peak throughput of these interfaces.

Most of the blocks in the data path contain some degree of data buffering, dictated by their own functional needs, with the bulk of the data buffering between the network and the host residing in the TX FIFO and RX FIFO blocks.

2.2.3.1 TX Data path

Figure 2-3 shows the width and speed of the TX Data Path between the different blocks.

Transmit data is transferred by the TX DMA block into the TX FIFO through the read buffers of the PCI interface block. The TX DMA and TX FIFO are capable of sustaining the maximum peak PCI burst rate for an entire packet transfer. The TX FIFO size is 9kbytes.

On the output side of the TX FIFO, data is transferred to the TX MAC using a 64- bit interface, nominally faster than the 1Gbps data rate of the network. The MAC layer, adds the relevant frame fields (Preamble, FCS) on the fly and generates a byte wide data stream (or nibble stream when using the MII).

The 8B/10B block encodes each byte into a ten bit code, which can be internally or externally serialized into the 1.25Gbps serial bit stream.

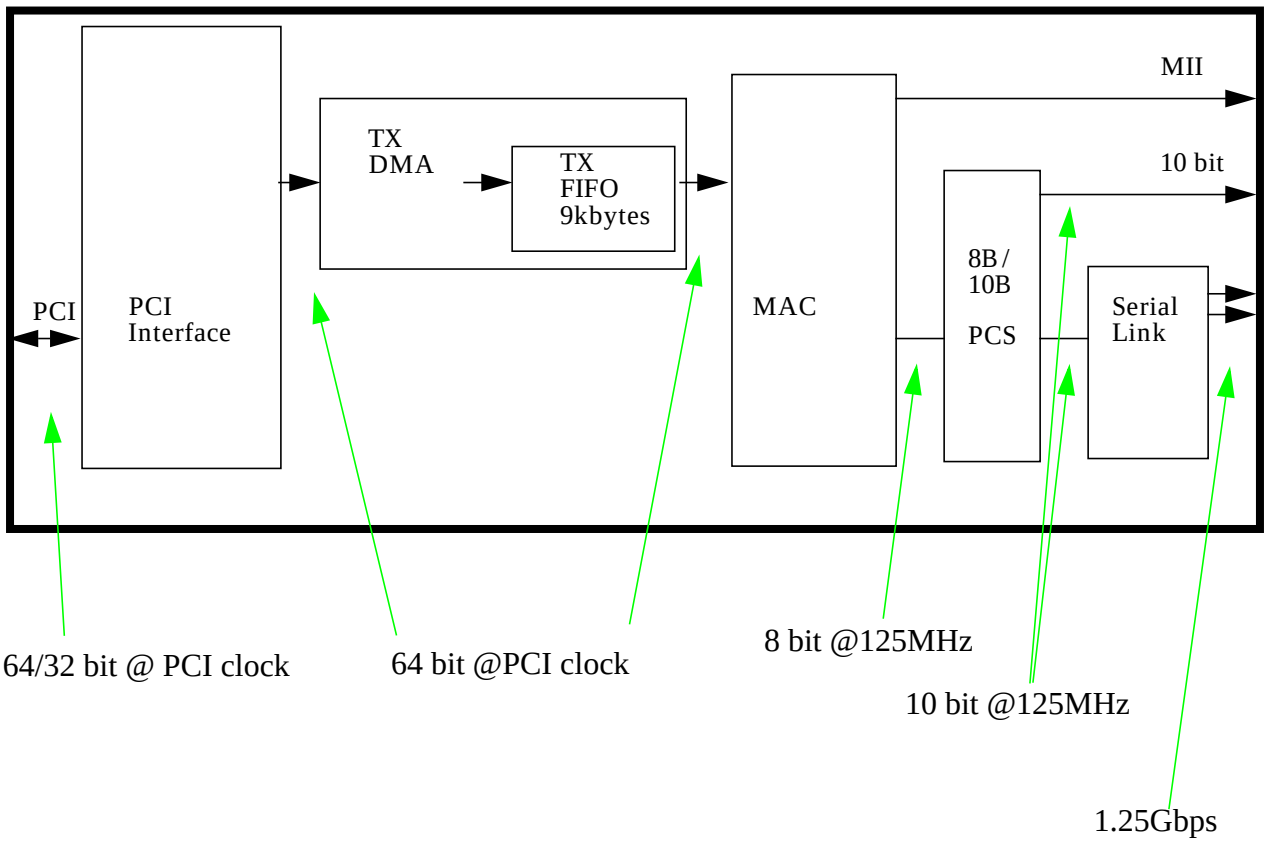


Figure 2-3 TX Data Path

2.2.3.2 RX Data path

Figure 2-4 shows the width and speed of the RX Data Path between the different blocks.

The 1.25 Gbps receive bit stream is presented either directly to the internal Serial Link module, or to an external SERDES chip. After clock recovery and bit alignment, the receive path is a 10 bit wide receive stream.

The 8B/10B block decodes it into a byte stream. The MAC is capable of receiving the bytes (or a nibble wide MII path) and after frame delineation, address filtering and FCS checking, pass 64-bit wide data, one frame at a time, to the RX FIFO. The RX FIFO size is 20kbytes.

The RX DMA will move the data to the PCI interface write buffer, while sustaining peak PCI burst DMA rates for entire packets if necessary.

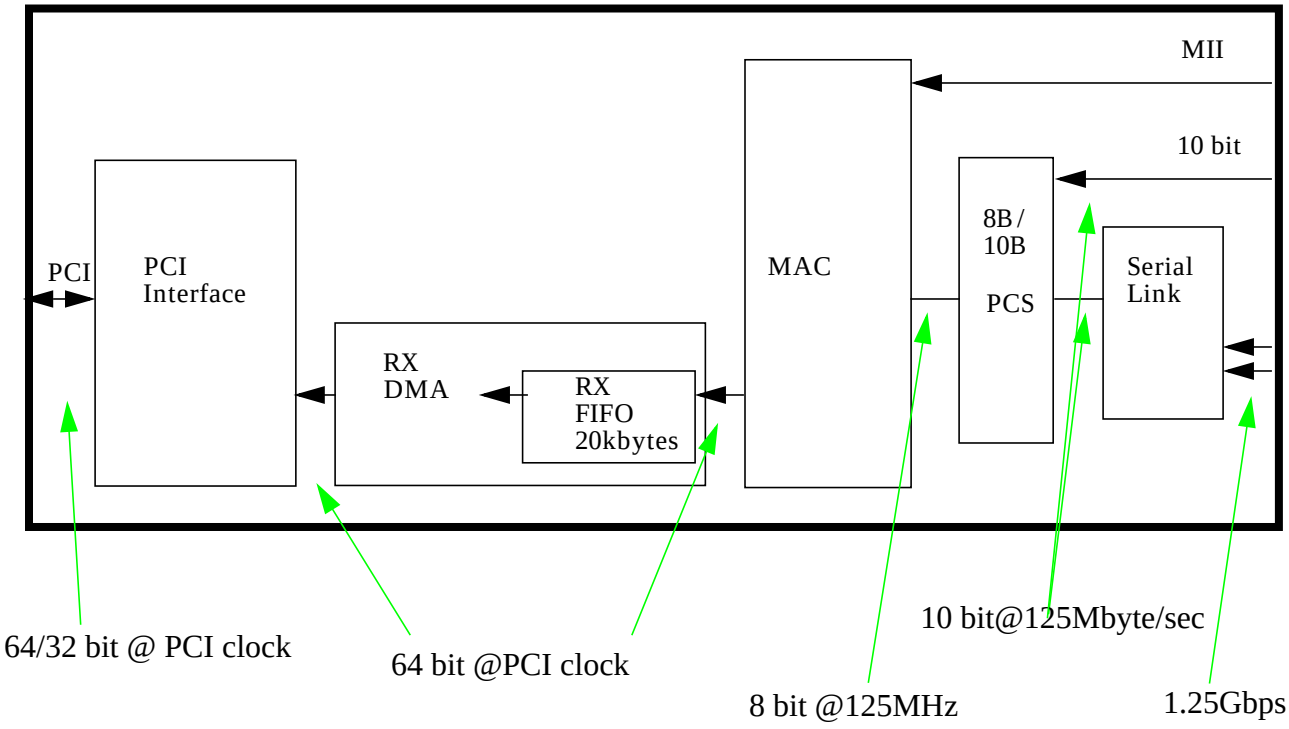


Figure 2-4 RX Data Path

2.2.4 MAC Control Frames

In addition to the usual Data frames, GEM supports MAC Control frames, as defined by IEEE 802.3x, for the purpose of implementing link level flow control. MAC Control frames are typically sourced and terminated at the MAC layer, and do not traverse the entire data path to and from the host interface.

MAC Control frames are distinguished by a specific value in the frame *type field*. The specific MAC Control function is determined by the value of the MAC Control Opcode field.

2.2.4.1 Handling of Received MAC Control Frames

On reception, MAC Control frames are decoded and acted upon by the MAC block itself. In the particular case of flow control frames, the MAC Destination Address may be a well known group address or the unicast address, and the MAC Control Opcode field is “*PAUSE*”. A MAC Control frame, like any other frame, must have a correct CRC to be considered valid.

GEM detects a “*PAUSE*” flow control frame by matching the Destination Address, the type field, and the opcode field to pre-programmed values, as well as verifying correct CRC. The duration of the requested PAUSE, is extracted from the frame and internally stored.

GEM’s pause timer is loaded with the PAUSE value, at the first opportunity after receiving a PAUSE frame that there is no transmission in progress at the MAC. The MAC may not transmit Data Frames while the pause timer is not zero. MAC Control frames are not normally passed to the system, but the pause timer as well as the last PAUSE value received are readable by the host.

2.2.4.2 *Sourcing of MAC Control Frames*

Given that MAC Control Frames are valid frames, they can certainly be sourced by the host as regular data frames, however this is not practical for lossless flow control due to the processing and queueing delays involved, and the fact that such MAC Control Frames, generated as Data Frames, would be delayed themselves by a non-zero Pause Timer condition.

The appropriate sourcing of flow control frames is internal to GEM. GEM can transmit MAC Control Frames that bypass the data path (i.e. do not go through the TX DMA to MAC interface). These frames are assembled by the MAC block using programmable values for Destination Address, Source Address, Type Field, MAC Control Opcode, and its “associated parameter” field.

For “PAUSE” Frames the host should program these fields with the values specified by 802.3x, and the associated parameter with the desired PAUSE timer value. GEM can autonomously send the pre-programmed PAUSE frame when the RX FIFO fills above a programmable OFF threshold. The PAUSE frame is emitted by the MAC at its next possible opportunity.

Whenever the RX FIFO usage falls below a programmable ON threshold, a similar PAUSE frame is emitted with the PAUSE timer value set to zero, allowing the sender to resume even if the original PAUSE timer had not expired yet.

GEM’s behavior is further constrained by the following guidelines:

1. After emitting an XOFF PAUSE frame, a second XOFF is not emitted unless the XON threshold was crossed.
2. XON frames will not be emitted unless the last PAUSE frame emitted was an XOFF.

Note – This alternating XON / XOFF scheme minimizes the number of PAUSE frames emitted. If the XON threshold is not crossed, it is possible for the sender to resume transmission and for the receiver not to emit additional XOFF PAUSE frames. For lossless performance the PAUSE timer value programmed in GEM must be large enough to ensure that the XON threshold will be crossed.

These two types of PAUSE frames can also be emitted, upon host request, by using GEM’s PIO write instructions. There are no restrictions on PIO triggered PAUSE frame emission.

PAUSE frames are triggered by the RX DMA block, but the enforcement of the above mentioned constraints, and the actual frame generation are done by the MAC according to the format shown in Table 2-1.

Table 2-1 Emitted PAUSE Frame Format

Field	Size	XOFF PAUSE frame	XON PAUSE frame
DA	6 bytes	Multicast Address defined for frame based flow control: 01-80-C2-00-00-01. Stored in MAC Address 8,7,6 Registers	
SA	6 bytes	Station Source Address, as stored in MAC Address 0,1,2 Registers	
Type Field	2 bytes	Value stored in MAC Control Type Register. Defined to be 0x8808	
MAC Control PAUSE Opcode	2 bytes	0x0001	
Associated Parameter	2 bytes	Pause Time in Slot Time units. (Pause Time is controlled via the Send Pause Command Register)	0x0000
PAD	42 bytes	0x00	
FCS	4 bytes	Frame CRC	

The PAUSE function is only specified by 802.3x for Full Duplex mode, and it is GEM's software driver responsibility to disable PAUSE frame generation for half-duplex operation.

2.2.4.3 *Frame Discard Policy*

Lossless flow control can be achieved as described in 2.2.4.2 as long as the sender implements flow control and the "PAUSE" emission threshold is programmed appropriately (typically up to two maximum sized frames may still arrive after the emission of "PAUSE").

A packet discard policy is necessary for cases where receive data overflows may still occur, for example, in half duplex mode, or in full duplex if the sender does not implement the flow control option. The discard policy in regards to the receive queues C and D of Figure 2-5 is as follows:

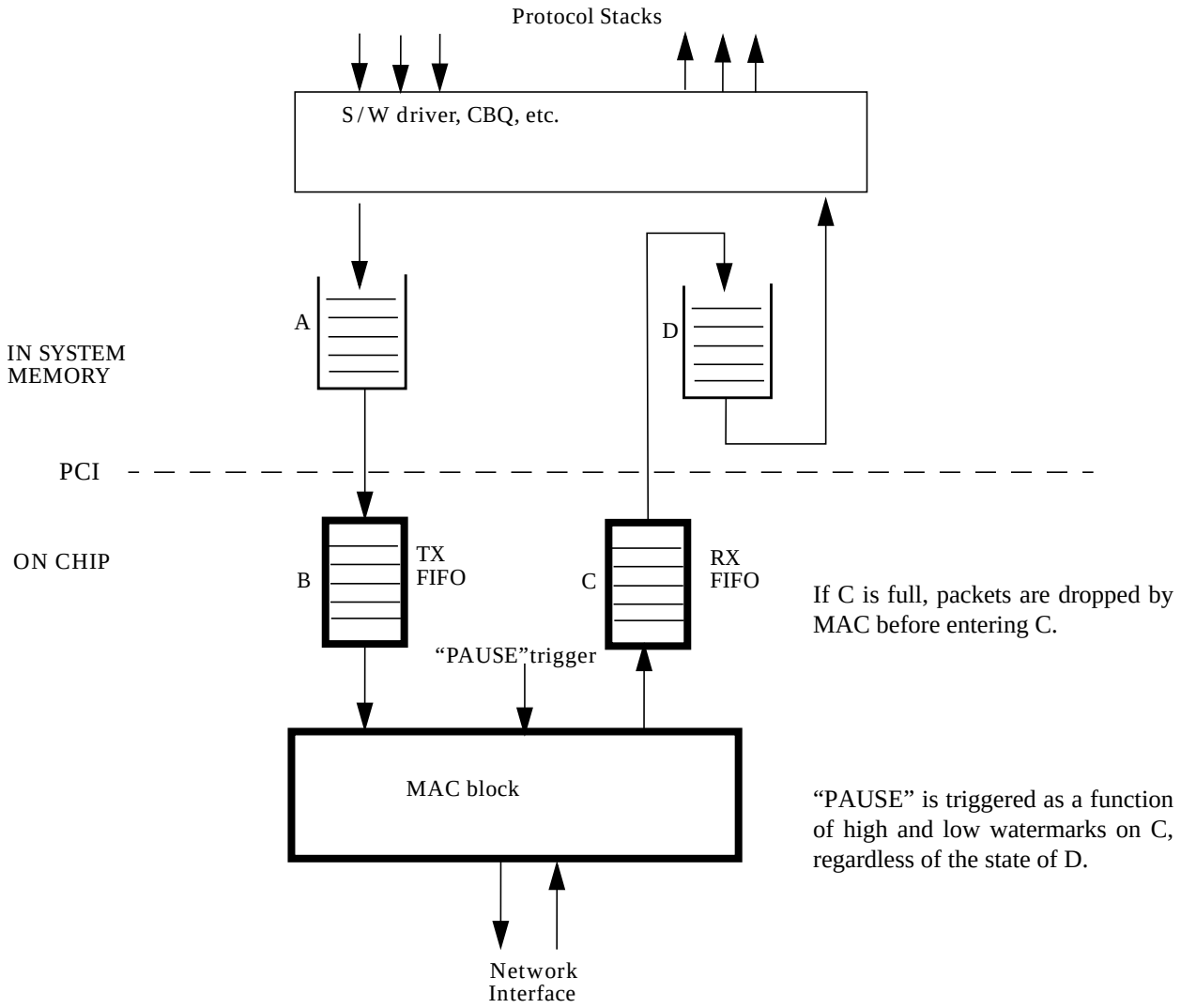


Figure 2-5 Frame Discard Policy

If queue D is full (no more free buffers in the receive system memory queue), GEM can generate an interrupt to notify the driver, but does not drop data if there is still room in the RX FIFO C.

If C fills up (either due to D being full, or just due to transfers between GEM and system memory being too slow), packet discards occurs at the input to C.

Once discard starts, it may only stop discarding on packet boundaries.

"PAUSE" frames are triggered as a function of C occupancy, regardless of whether D is full.

2.2.5 Data Structures

2.2.5.1 System Memory Data Structures

The transmit and receive queues in system memory, whose role was shown in Figure 1-3 on page 1-19, are implemented as “wrap-around descriptor rings”. These queues are jointly maintained by GEM and by the device driver. Given that the host frame processing time is likely to be dominant, and highly variable, large host queues are desirable in the descriptor rings. The number of entries is programmable in binary increments, from 32 to 8192 entries. Each descriptor consists of two double-word entries: a control/status entry and a pointer to a data buffer.

The interaction between the hardware and the software is managed using on-chip registers as well as the content of the descriptors themselves. Recognizing that several descriptors may occupy a cache line, and that descriptors typically reside in consistent memory space, GEM implements internal caching on descriptor reads to minimize the PCI and host overhead of repeated partial line reads and writes. Descriptor caching on reads applies to transmit and receive, and consists of fetching and internally storing up to four descriptors sharing a cache line. This has to be complemented by the software driver posting descriptors in groups of four, whenever possible. The equivalent technique for descriptor writes is descriptor “batching” and is covered below, in the *Receive Descriptor Ring* section.

Transmit Descriptor Ring

Figure 2-6 shows the Transmit Descriptor Ring organization. In addition to TX descriptors themselves, there are two registers involved in descriptor hand-over between hardware and software: the TX Kick Register and the TX Completion Register.

A transmit frame that is posted by an upper layer protocol to the device driver may reside in several data buffers (headers and data), which are scattered in the system memory. When the device driver posts the frame to the hardware, it allocates one descriptor in the Descriptor Ring for each buffer. The descriptor contains the information necessary for the hardware to transfer the frame data from the data buffer.

When the frame is ready for transmission, the descriptor is queued to the hardware, by writing a new value into the TX Kick Register.

When a buffer transfer has been completed, the transmit DMA channel turns over the descriptor ownership back to the driver by updating the TX Completion Register. The transmit DMA engine will continue processing descriptors and transfer frames from the data structures that precede the last queued descriptor as reflected by the TX Kick Register.

Note that in order to avoid GEM writing back descriptors, there is no explicit ownership information in the descriptor. Descriptor ownership information is derived from the values of the TX Kick Register and TX Completion Register.

For Transmit, descriptor caching reduces bus overhead by reading up to four descriptors in the same burst, however the software driver cannot guarantee that it will post more than one descriptor at a time.

The TX DMA may only internally cache descriptors if the TX Kick Register indicates that there is more than one descriptor to be fetched within the same cache line. Otherwise a single descriptor is read.

The maximum number of outstanding transmit descriptors that GEM can handle is given by the transmit descriptor size minus one.

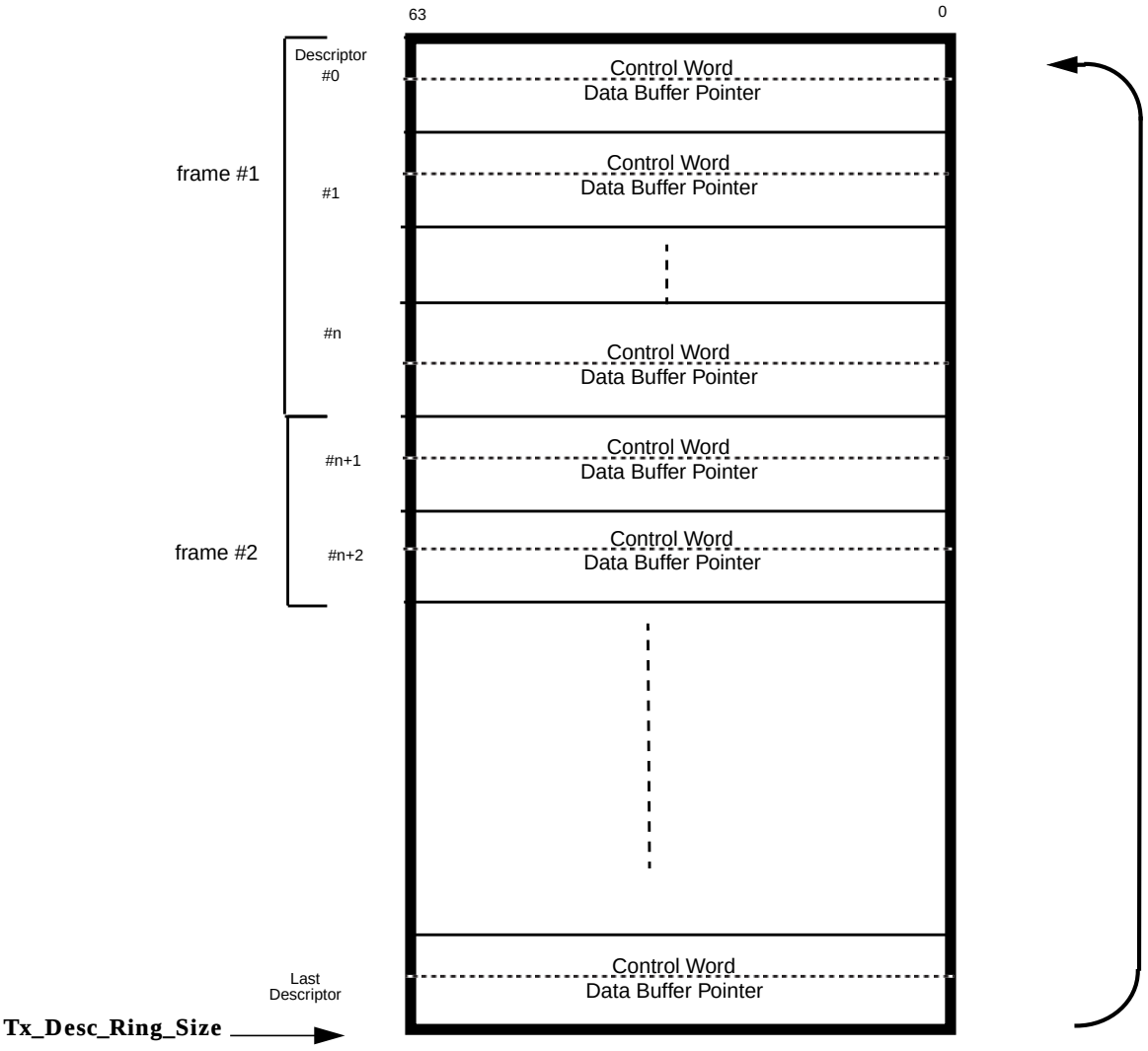


Figure 2-6 Transmit Descriptor Ring Organization

TX Descriptor Entry Layout

63 34	33	32	31	30	29	28 21	20 15	14 0
RESERVED	No CRC	Int Me	Start of Frame	End of Frame	Check sum Enable	Checksum Stuff Offset	Checksum Start Offset	Buffer Size
Data Buffer Pointer								

TX Descriptor Fields

All TX Descriptor fields are written by the host and read by GEM. GEM never writes into TX Descriptors.

Data Buffer Pointer - Pointer to the first data byte of the transmit buffer.

Buffer Size- Number of data bytes in the buffer. Valid values are in the range 0 to 17k.

Checksum Start Offset - Number of bytes from the first byte of the packet to be skipped before checksum calculation begins. Value must be even. Only relevant for the first buffer of a packet with Checksum Enable bit set.

Checksum Stuff Offset - Indicates the byte offset within the packet of the first checksum byte. Value must be even. Only relevant for the first buffer of a packet with Checksum Enable bit set.

Checksum Enable - When set to ‘1’ for the first descriptor of a packet, checksum is stuffed for the given packet.

End of Frame - When set to ‘1’ it indicates the last descriptor of a packet.

Start of Frame - When set to ‘1’ it indicates the first descriptor of a packet.

Int Me - When set to ‘1’ in the first descriptor of a frame, the TX_INT_ME interrupt will be set upon loading the entire frame into the TX FIFO.

No CRC - CRC is not inserted to packets with this bit set in their first descriptor.

Receive Descriptor Ring

Figure 2-7 shows the Receive Descriptor Ring organization.

For receive operation, the device driver requests free buffers from the Operating System. The buffers are posted to the hardware by allocating descriptors in multiples of four, one descriptor corresponding to a buffer. The descriptor contains the buffer information necessary for the frame transfer. Unlike transmit, receive descriptors include the OWN bit that may be used by the CPU to determine descriptor hardware vs. software driver ownership.

Whenever frame data is ready to be transferred from the RX FIFO to system memory, the receive DMA determines if a descriptor is already cached inside the device. If none is available but the RX Kick Register value indicates that additional valid descriptors are available in the descriptor ring, four new descriptors are fetched. Data transfer begins into the buffer pointed to by the next cached descriptor.

When the frame transfer has been completed, the receive DMA channel updates the descriptor with status information about the received frame, and turns over the descriptor ownership back to the driver, while generating an interrupt.

If the driver does not have any free buffers allocated to the hardware, it indicates this by generating an interrupt. Receive data packets continue to be received and buffered internally for as long as the RX FIFO space allows. The DMA channel will resume its transfers only after the value of the RX Kick Register indicates that free buffers are available.

If receive packets arrive far apart GEM updates the RX descriptors one at a time (partial line write). If receive packets arrive at a higher rate, GEM will update four descriptors at a time, potentially writing entire cache lines.

Given that the software driver must post descriptors in multiples of four, and that the ring should not wrap around, the number of outstanding descriptors cannot exceed the descriptor ring size minus four.

RX Descriptor Entry Layout

63	62	61	60	59 44	43 32	31	30 16	15 0
Reserved	Bad	Alternate	Hash Pass	Hash Value	Reserved	Own	Frame Data Size	TCP Pseudo-Checksum
Free Buffer Pointer								

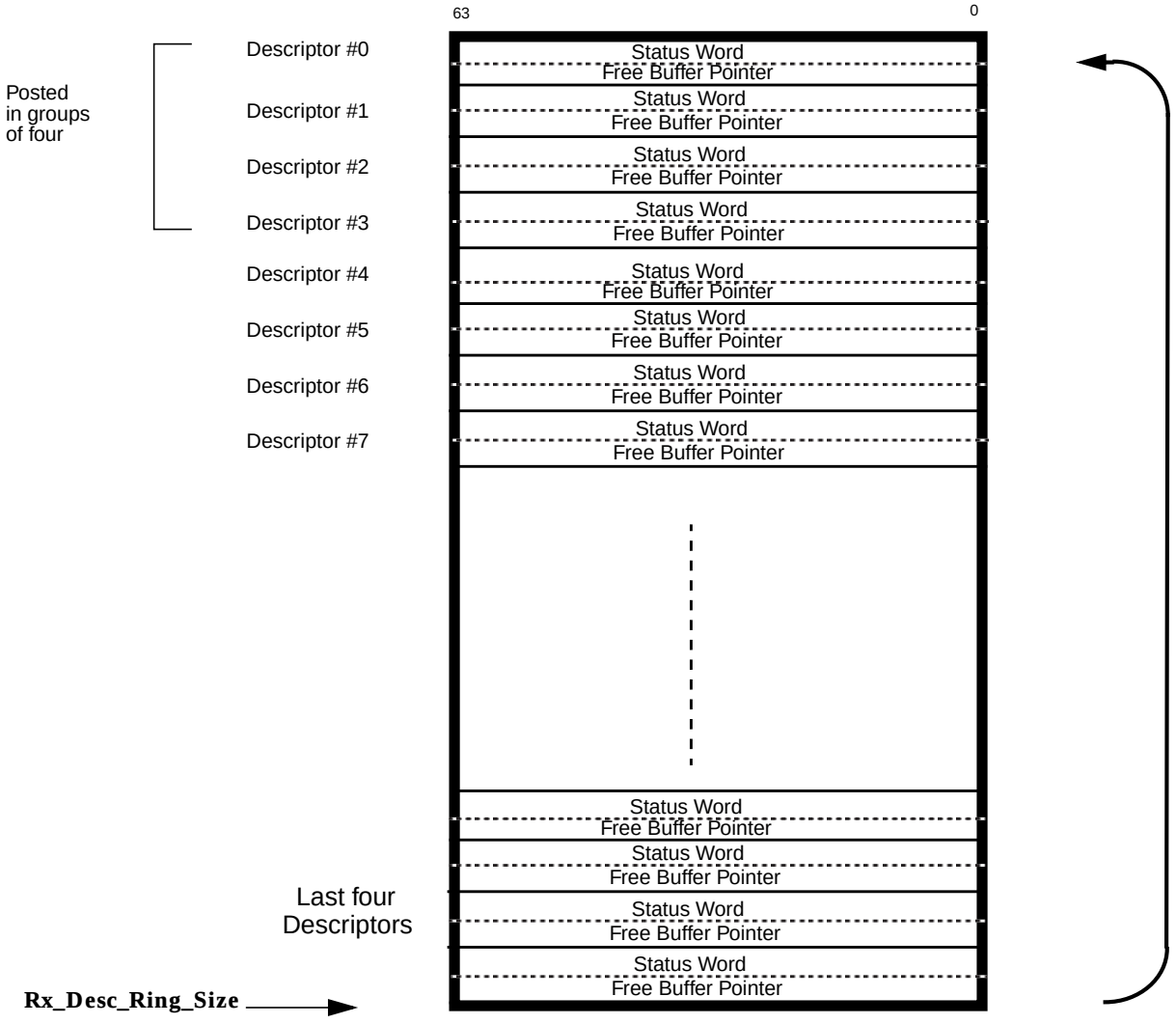


Figure 2-7 Receive Descriptor Ring Organization

RX Descriptor Fields

The Free Buffer Pointer is determined (written) by the host. GEM reads it, and writes back its value intact when writing the RX descriptor.

Free Buffer Pointer - Pointer to the beginning of the free buffer. The three least significant bits are ignored. The first actual data byte will be at an offset from this address, of up to eight additional locations. The offset is programmable in the RX Configuration Register.

The following RX Descriptor fields are written by GEM upon packet reception, and are ignored by GEM when reading a descriptor.

TCP Pseudo-Checksum - Contains the 16-bit TCP checksum calculated over the entire frame, starting at an offset programmable in the RX Configuration Register.

Frame Data Size - GEM writes this field to indicate the number of data bytes in the frame.

OWN - Ownership semaphore. GEM clears this bit when writing the descriptor. This field may assist the CPU in determining ownership, but it is not checked by GEM.

HASH VALUE - This field provides the hash value of the Destination Address and may be used by the host to accelerate its own perfect filtering function.

HASH PASS - When set, this bit indicates the received packet destination address matched the hashing filter criteria.

ALTERNATE - When set, this bit indicates the received packet destination address matched the alternate MAC address stored in MAC Address Registers 3, 4, and 5.

BAD - Indicates the frame is a bad frame (BAD CRC). May be used by the software driver to identify bad packets when configured to receive bad packets.

Checksum Considerations

The checksum is assumed by GEM to be a 16-bit field used by some upper layers, like TCP, to verify the integrity of frames received. Given that checksum is not defined at the MAC level, the details of checksum generation/checking vary as a function of the specific upper layer sourcing or consuming the frame. GEM's checksum algorithm is specific to the TCP transport layer protocol.

For receive frames, a checksum is computed on even byte boundaries, and made available to the host as part of the frame's RX Descriptor. The checksum covers the frame data fields starting at a programmable offset, normally excluding the FCS, except when GEM is instructed to store FCS. The host must determine if the receive checksum is relevant for each frame's upper layer protocol, and compensate for the header bytes and the even byte alignment, if necessary. Receive frame checksum is always computed by GEM, and is optional in the sense that the host can selectively ignore it whenever it does not need it, or can compute it faster in software.

Transmit frame checksum generation affects the frame format on the cable, therefore the checksum coverage and placement within the frame are rigidly defined for a given upper layer protocol. These two are supplied by the host, on a frame by frame basis, as parameters in the TX descriptor. GEM restricts “Checksum Start Offset” and “Checksum Stuff Offset” to 16-bit alignment (the lsb of these parameters is forced to zero). Transmit checksum is optional in the sense that it will not be inserted by GEM unless specifically enabled in the TX Descriptor of the frame.

2.2.6 PCI DMA functions

Data and descriptor movement to and from host queues is done by GEM as a PCI DMA bus master. As a Gigabit full duplex device, it is important to optimize the bandwidth of GEM’s memory access functions, with the characteristics of Sun’s PCI platforms in mind.

A factor already presented in “System Memory Data Structures” is that the TX descriptor writeback is normally eliminated by the inclusion of TX Completion Register.

As for the actual DMA behavior, when allowed by the DMA target and the arbiter, GEM is capable of bursting entire frame lengths to maximize efficiency on the bus. This capability is achieved by large FIFOs, and by having enough internal bandwidth between the PCI and the FIFOs to sustain such long bursts.

To minimize receive latency, it might be beneficial to program the receive DMA to start transfers before a full frame is received. In any case DMA requests are only issued when entire cache lines are ready to be transferred, with the only exception being end of frame transfers.

The following section define the functional guidelines for GEM’s DMA on PCI:

2.2.6.1 DMA Data accesses

a. Burst entire cache lines for DMA writes (for sub-line writes, use garbage fill to avoid partial writes)

b. Burst just the relevant data for DMA reads

c. Start bursts on cache line boundaries

d. Use MWI for writes, except when starting unaligned (after retry, etc.), or when the *Cache Line Size* is zero or an unsupported value (larger than 128 bytes or non power of 2).

e. If latency timer expires and GNT removed, MWI should complete writing the current cache line before relinquishing the bus.

2.2.6.2 DMA Descriptor accesses

a. TX DMA reads up to four descriptors per burst (if available within same cache line) and caches them internally

b. RX descriptor always reads four descriptors and caches them internally

c. No TX descriptor writes

d. RX descriptor writes use MW for sub-line writes

e. RX descriptor “batching” may use MWI to write up to four descriptors in a cache line

2.2.6.3 DMA arbitration and bursts

a. Support cache line bursts and infinite burst modes

b. If the cache line value is zero, or not supported by GEM, 64 byte bursts are used instead of cache line bursts, and no cache line commands are issued on PCI

c. Start TX read burst, or continue the burst into the next cache line only if the entire cache line can be DMA’ed (except at the end of a buffer).

d. Start RX write burst, or continue the burst into the next cache line only if the programmable RCV DMA threshold is satisfied and the entire cache line can be DMA’ed (except at the end of a frame).

e. GEM should be ready to transfer data on every burst clock (IRDY# low at all times)

f. Always use linear increment addressing mode (AD[1:0]=00)

g. Fast back-to-back transactions from different agents (as a target) are not supported

2.2.6.4 Internal Arbitration

The presence of the TX and RX FIFO allows systems that support long DMA bursts on PCI to increase bus transfer efficiency by minimizing the need to switch between TX and RX DMA channels too frequently. The “Infinite_Burst” bit in the Configuration Register determines whether DMA transfers are broken into cache line transfers, or whether the active channel is allowed to continue for up to an entire packet.

The “TX_DMA_Limit” and “RX_DMA_Limit” values programmed in the Configuration Register are meant to address the side effects of infinite burst DMA, when both channels are active (increased latency, bandwidth mismatch due to packet size mismatch).

Programming different DMA_LIMIT values for the TX and RX channels also constitutes a mechanism for controlling the relative ratio of the bus that each channel will use during periods of simultaneous TX and RX DMA activity. The DMA_LIMIT is re-applied every time the channel acquires the bus, and does not accumulate over more than one bus acquisition.

The following table shows the internal arbitration behavior as a function of the value programmed as the DMA_LIMIT of a given channel:

Table 2-2 Internal TX vs. RX arbitration

DMA_LIMIT (of the channel granted)	DATA TRANSFER BOUNDARY	PRIORITY
k	If the other channel has no pending requests the burst may continue up to the end of the packet. Otherwise stop on first cache line boundary after at least (k * 64) bytes data transfers.	Round-robin, set to opposite channel upon transfer of (k * 64) bytes

2.2.6.5 DMA address generation

a. Capable of generating 64 bit addresses for descriptors and data

b. Only the lowest 41 address bits are manipulated for pointer generation, the higher 23 bits are taken as is from the appropriate register / descriptor

c. Dual Address Cycles are used when the upper 32 address bits are non-zero

2.2.7 Endianess

As a PCI device, GEM follows a little endian bus organization for all its blocks, spaces and transfer types.

2.2.8 Interrupt Behavior

GEM uses a single interrupt line to signal events that require host intervention. The specific interrupt events are defined by a two-level interrupt status structure. The main events have their corresponding bits in the interrupt status register, while secondary events are reported by secondary status registers whose bits share an entry in the interrupt status register.

Enabling (masking) and clearing of every interrupt condition is done at its own status register level. No additional restrictions exist for polling the interrupt bits at any time. The main interrupt status register may be read through two locations, one of them automatically clears all its active interrupts upon reading it, while the other location allows reading with no side effects. Secondary status registers are cleared upon reading them.

The receive interrupt is the main condition used by the software driver to remove frames from the host receive queue and pass them to the upper layers. If the receive frame arrival rate lags behind the frame processing rate, the receive interrupt will be activated for every packet, subject to a blanking mechanism. If the receive frame arrival rate is faster, the interrupt handler may process more than one frame per interrupt. The receive interrupt blanking mechanism may be used to delay the receive interrupt assertion until a programmable number of receive packets were received, or a minimum time has elapsed since the last receive interrupt.

Transferring frames between the upper layer and GEM is not interrupt driven, however transmit interrupts may be enabled to assist in the scheduling and buffer reclaim of the transmit process. Transmit interrupts may be requested on any arbitrary frame(s) using the TX_INT_ME event, and will be generated when the specific frame(s) is completely transferred from the host queue to the TX FIFO.

The interrupt structure is defined with the guidelines of:

1. Minimizing dependencies between receive and transmit activities

2. Minimize number of slave reads required per interrupt

The value of the TX Completion Register is also made available in the Interrupt Status Register space to allow the software driver upon any interrupt event to quickly determine how many transmit buffers, if any, it may reclaim. This field is not an interrupt source in itself.

Several other interrupts conditions exist. Some of them are associated with error conditions, maintaining statistics, or reflecting a change in the state of the link or transceiver. The Receive Buffer Overflow interrupt indicates the reception of a frame that was longer than the receive buffer allocated by the driver.

The interrupt status structure is illustrated in Figure 2-8. The detailed interrupt conditions and register definitions are specified in the Programmer’s Model Chapter.

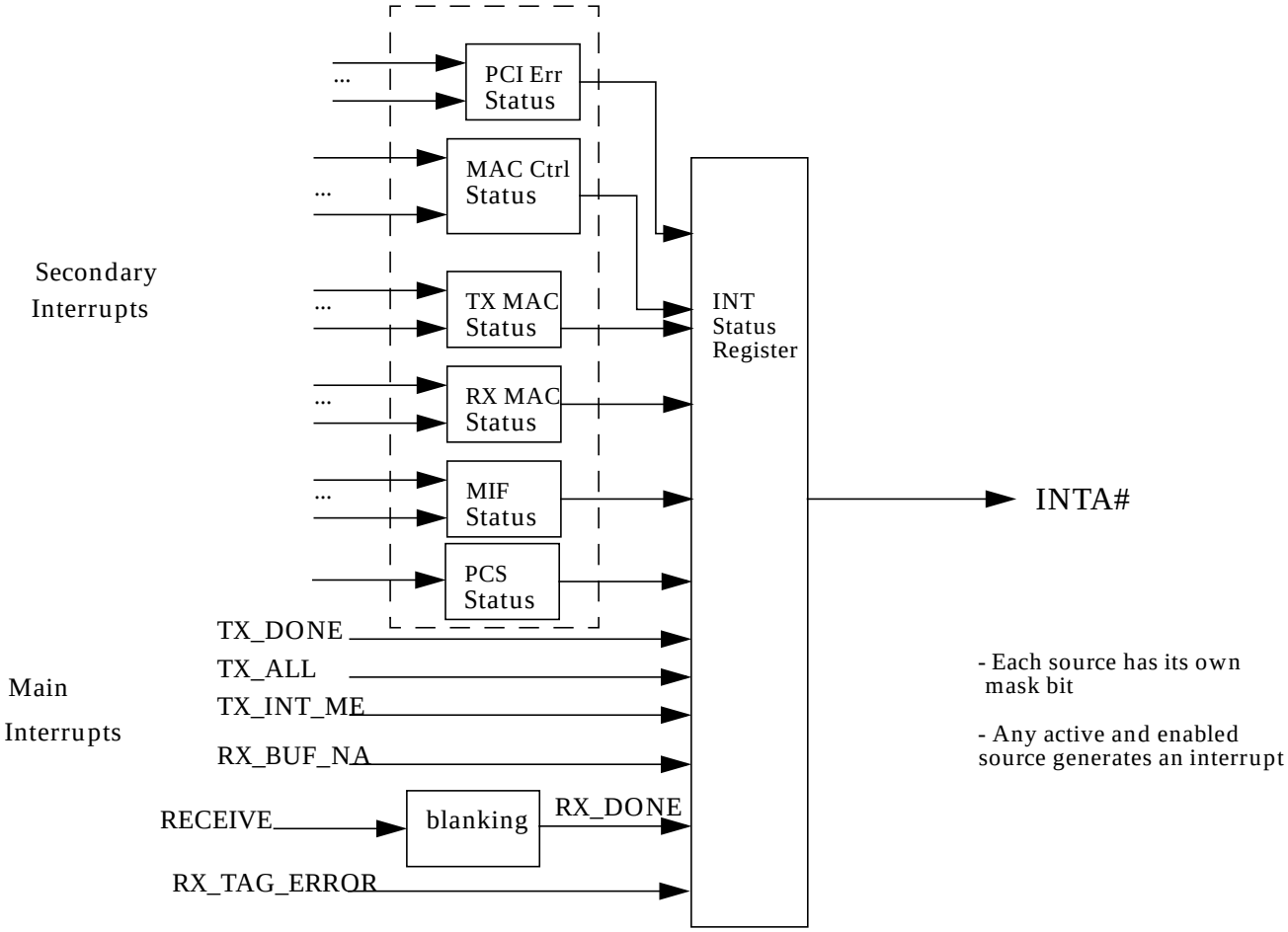


Figure 2-8 Interrupt Structure

2.2.8.1 Receive Interrupt Blanking

The RX_DONE receive interrupt can only be set if at least one of the following conditions is true:

1. The number of packets received since the last time RX_DONE was set is at least RX_INTR_PACKETS.
2. The time elapsed since the last time RX_DONE was set is at least RX_INTR_TIME.

The values of RX_INTR_PACKETS and RX_INTR_TIME are programmed in the RX Blanking Register. RX_INTR_TIME is specified in units of PCI clocks divided by 2048.

2.2.9 Network Interface

The network interface functions of GEM provide the mechanisms for properly forming frames, following the protocols for when to send such frames, accepting frames, and allowing GEM to present only the relevant frames and control information to the host. There is significant flexibility in the parameters that control these mechanisms, including network speed, access method, framing fields lengths and values.

It should be noted that all these network interface options have no impact on the data structures presented to the host, or the internal interface between the MAC and TX and RX FIFOs. Some of these options do affect the interface between GEM and the external transceiver circuitry necessary to interface the network.

2.2.9.1 100BASE-T and 10BASE-T Networks

For 100BASE-T networks, GEM preserves the functional partitioning at the MII level. GEM implements all MAC layer functions above the MII in half and full duplex modes. 100Mbps (as well as 10Mbps) connectivity can be implemented using commercially available MII transceiver chips.

2.2.9.2 Gigabit Networks

The natural interface to the anticipated Gigabit links, in particular for fiber transceivers, is based on leveraging Fiber Channel type transceivers. Both FC-0 (Physical) and FC-1 (Code) Fiber Channel are applicable for Gigabit fiber links, but the leverage is directed mostly at using FC-0 devices, while incorporating the FC-1 function on chip.

GEM provides a built-in PCS (Physical Coding Sublayer) with its corresponding 8B/10B coding. The PCS functions may be used in conjunction with external SERDES chips or Fiber Channel transceivers via a standard 10-bit interface, or by using GEM's own built-in (PMA) serializer/deserializer and clock recovery circuit. The external interface in this case is a serial PECL differential pair at 1.25Gbps.

When using external PMA and PMD components, note that ANSI X3T11 specifies 1.0625 Gbps operation, and GEM requires transceivers capable of 1.25Gbps bit rate.

An additional interface is the GMII. It represents an extension of the MII to accommodate 1Gbps operation for non-8B/10B media interfaces.

In order to minimize the pinout impact of these interfaces, the MII, GMII, and SERDES interfaces share many pins, and typical GEM boards should only populate one interface.

2.3 Theory of Operation and Data Flow

2.3.1 Frame Transmission

Host Memory to TX FIFO Data Transfer Process

1. Frame transmission is initiated by an upper layer protocol posting a frame to the transmit queue in host memory. The frame may reside in one or more data buffers.

2. The device driver allocates a descriptor for each buffer in the corresponding transmit descriptor ring.

3. The device driver writes into the TX Kick Register the descriptor number that follows the last valid descriptor.

4. Upon detection that the value of the TX Kick Register is ahead of the TX Completion Register, the TX DMA arbitrates for access to the PCI bus. The arbitration between the TX and RX DMA is done by the PCI Interface block.

5. The TX DMA engine uses a DMA burst read and fetches a number of descriptors given by MAX [valid new descriptors in the cache line , 4]. The descriptors read are cached internally for use in the order they were fetched.

6. The TX DMA starts transferring the contents of the corresponding host data buffer into the TX FIFO, and advancing the Write pointer accordingly. The DMA read data transfer attempted is always a burst of an integer number of cache lines, except possibly at the end of the frame transfer if less than a full line remains.

7. On its way to the TX FIFO, the data passes through the Chaining block where byte alignment is performed. For the first host data buffer of a frame, the data is aligned to a double-word boundary in the TX FIFO. For subsequent buffers, “byte rotation” is performed to form a contiguous data stream in the TX FIFO for each frame.

8. As the “chained” data stream “flies by” on its way to the TX FIFO, the Checksum block monitors it, and the TCP checksum is computed starting from the “start_checksum_offset” byte. Also, the word indicated by the “stuff_checksum_offset” is saved in a temporary holding register.

9. Upon completion of a host buffer transfer, the TX DMA engine turns over the descriptor ownership back to the device driver by writing into the TX Completion Register the descriptor number that follows the last descriptor processed.

10. If the frame contains more than one host data buffer, and descriptors are already cached internally, steps 6 through 9 are repeated until the entire frame has been transferred from the host memory to the TX FIFO. If at any point there are no more cached descriptors for the frame, steps 4 through 9 are repeated instead.

11. If the TCP checksum generation for the frame is enabled, the computed checksum is stored into the holding register, and the updated word is loaded in the frame header in the TX FIFO using a programmable offset from the beginning of the frame as reflected by the Shadow Write pointer.

12. The TX DMA assembles a Control Word and appends it to the end of the frame in the TX FIFO. The Control Word indicates the last byte boundary of the frame, and indicates to the TxMAC whether to generate CRC for the frame. The TX FIFO Write Pointer is loaded into the Shadow Write Pointer and point to the beginning of the next frame in the TX FIFO.

13. Two interrupt mechanisms can report frame transfer completion at this point. The TX_DONE interrupt status bit is unconditionally set, and the TX_INT_ME status bit is set only if requested in any of the descriptors of the frame.

14. The TX FIFO packet counter increments for every frame written into the TX FIFO.

TX FIFO to Network Data Transfer Process

1. This process is triggered by the presence of at least one full frame (TX FIFO packet counter is not zero) in the TX FIFO, or by the number of bytes exceeding a programmable threshold. The TX FIFO read pointers (Read Pointer, Shadow Read Pointer) point to the beginning of the frame in the TX FIFO.

2. On the output of the TX FIFO, the frame is unpacked into a 64-bit data stream, and is transferred to the TxMAC in bursts of 32 bytes. The Read Pointer is incremented reflecting the last TX FIFO location read.

3. Should the TX FIFO be empty during frame transfer, data transfer to TXMAC stalls until more data becomes available in the TX FIFO.

4. The TX DMA relies on the TxMAC for discriminating between normal and late collisions. Upon normal collisions, the frame is re-transmitted by loading the Shadow Read Pointer into the Read Pointer. No retransmission is requested by TxMAC upon late collisions.

5. TX DMA assumes that normal collisions cannot occur after having transferred 1kbytes. Beyond that point the Shadow Read Pointer tracks the Read Pointer.

6. When the last data word (distinguished by the tag bit set) appears in the TX FIFO, the following word (status word, with the tag bit set as well) must also be examined to determine how many bytes are valid in the last word, and encode that information during the last data transfer to the Tx MAC.

7. The TX FIFO packet counter decrements for every frame read from the TX FIFO when the TxMAC requests the next packet. This guarantees that the current packet does not need retransmission.

2.3.2 Frame Reception

Network to RX FIFO Data Transfer Process

1. When a frame is received and it satisfies the RxMAC filtering criteria, the frame will be passed by the RxMAC to the RX FIFO. The Write Pointer and Write Shadow Pointer coincide and point to the next RX FIFO location.

2. The data is packed into 8-byte words by the RxMAC. Transfers take place whenever the RxMAC has at least 8 bytes to transfer and the RX FIFO has room for them. The Write Pointer increments on these transfers while the Write Shadow Pointer still points to the beginning of the frame.

3. As the frame data stream “flies by” on its way to the RX FIFO, the Checksum block monitors it, and the TCP checksum is computed on the entire MAC frame, starting from a 16-bit aligned programmable “start_checksum_offset”.

4. The data is packed on its way to the FIFO and the first byte is aligned considering the three least significant bits of the programmable receive start offset. Byte rotation is performed for the remainder of the packet.

5. After the entire frame has been transferred to the RX FIFO, indicated by the arrival of a status word from the RxMAC, the status word is modified to include the TCP checksum. The frame length is provided by the Rx MAC as part of the status word.

6. The last word as well as the status word are written into the RX FIFO with the tag bit set.

7. If at any point during this process the RX FIFO occupancy rises above the OFF threshold, a “XOFF PAUSE” frame transmission request to the Tx MAC may be triggered, as described in Section 2.2.4.2, “Sourcing of MAC Control Frames”. The Tx MAC may ignore the request if it is not configured appropriately (Send_Pause_Enable must be set).

8. If the need arises to discard the frame in progress, either because of RX FIFO congestion or some frame error, the Write Pointer may be rewound to the value of the Write Shadow Pointer only if the RX DMA has not already started reading the frame. Namely if either the RX FIFO packet counter is more than 1, or less than max [RX_DMA_Threshold , cache line size] data bytes of the frame were already written into the RX FIFO. See 2.6.1 for details on frame abort.

9. For every frame written into the RX FIFO the RX FIFO packet counter is incremented.

RX FIFO to Host Memory Data Transfer Process

1. If at least one descriptor is internally cached when the RX DMA needs to move Data to Host Memory, it proceeds as described in step 3. If no descriptors are internally cached it proceeds with step 2.

2. If the values of the RX Kick register and the RX Completion register are equal, the RX DMA generates an interrupt (Rx_Buf_NotAv). Only when the values of the RX Kick register and the RX Completion register are different, the RX DMA engine fetches the next four descriptors from host memory using a DMA burst read, and proceeds to step 3.

3. RX DMA starts transferring the frame data from the RX FIFO to the host buffer. Except for the residual transfers at the end of a frame, these data transfers are only started when the number of bytes in the RX FIFO exceeds both the RX_DMA_threshold programmable value and the cache line size, or when the RX FIFO packet counter is not zero.

4. For the initial transfers of a frame (less than a cache line), the RX DMA performs “header padding” by inserting the number of “junk bytes” required at the beginning of the cache line to match the programmable start offset.

5. The frame transfer is completed when two consecutive tag bits have been detected by the RX DMA. If the status word indicates the packet was valid the RX DMA engine updates the registers that hold the descriptor contents with the appropriate values for the frame and clears the OWN bit.

6. Up to four sets of descriptors can be internally stored before an actual descriptor write is initiated on the bus. If batching is enabled, the descriptor number, and the RX FIFO packet counter are used to decide whether to update a descriptor immediately or to collect four descriptors and then update them all (“batching”). Descriptors are written whenever the fourth descriptor is ready, or the RX FIFO packet counter indicates no more packets await in the RX FIFO.

7. After using any descriptor for a new packet and updating it in memory, an interrupt is generated to indicate to the device driver the successful completion of the frame transfer. The value of the RX Completion Register is incremented by the number of Descriptors written (modulo RX Descriptor Ring Size) prior to setting the interrupt.

8. For every frame removed from the RX FIFO the RX FIFO packet counter is decremented.

9. If at any point during this process the RX FIFO occupancy falls below the ON threshold, a “XON PAUSE” frame transmission request to the Tx MAC may be triggered, as described in Section 2.2.4.2, “Sourcing of MAC Control Frames”. The Tx MAC may ignore the request if it is not configured appropriately (Send_Pause_Enable must be set).

2.4 Internal Functional blocks

This section describes in further detail the role and functional behavior of the blocks illustrated in Figure 2-1.

2.4.1 PCI Bus Interface (BIF) block

In addition to the PCI related DMA guidelines described in 2.2.6, the PCI interface block is defined with the following guidelines:

1. Bus width and speed determination
   a. optimize 64 bit transfers using a configuration bit, while still monitoring ACK64#
2. GEM never issues
   a. LOCK#
   b. Interrupt acknowledge, special cycle, I/O cycles, Configuration cycles
   c. Address or data stepping
3. Slave Accesses
   a. 32 bit wide Register Space and Configuration Space
   b. Bounded access time to internal registers
   c. Avoid sharing resources that may be used by different independent processes/CPUs

2.4.1.1 Slave PCI Accesses

The detailed register mappings for GEM's slave address spaces are described in Chapter 3, "Programmer's Model". The three slave access spaces are:

PCI Configuration Space

This is a 256 byte address space that follows the format of a single-function device. It is mostly a read/write space accessible as bytes, half-words or words, and must be accessible at all times. Values programmed into PCI Configuration space are not to be affected by software reset, and only return to their default values upon hardware reset.

PCI Configuration registers are implemented in the PCI block. Their decode starts at offset zero and is enabled by the IDSEL input (per slot addressing).

GEM Register Space

Decodes for various blocks in the chip are generated by the PCI Interface block. The base address for this space is determined by the Base Address Register in PCI Configuration Space. Accesses to GEM's registers is limited to 32 bit word accesses, and is mapped as non-prefetchable PCI memory space.

Expansion ROM Space

The Expansion ROM Space is a read-only space typically accessed by the system during POST. While the decoding of this space is done by GEM, the actual data is stored in the byte wide external PROM. GEM performs byte stacking to support any combination of 8-bit, 16-bit and 32-bit accesses. The Expansion ROM Base Address Register in PCI Configuration space is used to define the base address using bits [31:20], while bits [19:11] are hardwired to zero to define a 1Mbyte space. The decoding of the Expansion ROM is enabled by bit 0 of this register and the Memory bit of the Command Register being both set. Physically GEM supports up to 64kbyte PROMs, mapped at the beginning of this 1Mbyte space.

The Expansion ROM is also readable during run-time via the Register Space.

The delay introduced by byte stacking is handled in two possible ways:

- Accesses via the Expansion ROM space will complete after all required data bytes are read into GEM. The PCI target initial latency rule does not apply in this case.

- For Accesses via Register space, if the byte stacking delay exceeds the target initial latency GEM will terminate the cycle with a retry, and will continue prefetching up to 4 bytes in preparation for the cycle being retried by the initiator.

The PROM access timing is fixed at 10 EPCI clocks (5 PCI clocks).

2.4.2 TX DMA block

The TX DMA block works with the PCI BIF block to read transmit frames from the host memory. The DMA block handles the chaining of TX frames and TCP checksum generation. The TX DMA attempts to move data on cache line boundaries between the PCI BIF block and the TX FIFO. Handling of misaligned bursts due to PCI retries is handled within the PCI BIF and is transparent to the TX DMA.

Chaining

The TX DMA block implements the gather function of transmit buffers. Proper byte alignment is performed to ensure that data bytes at multiple buffers boundary, belonging to the same frame, are packed sequentially in the TX FIFO. The chaining of transmit buffers is done on-the-fly during frame data transfer between the PCI Bus Interface and the TX FIFO.

Checksum

This block provides the hardware support for TCP checksum computation in the transmit data path. This optional function can be enabled or disabled on a per frame basis through a descriptor control bit. If the function is enabled, the software has to provide, in the descriptor, a start offset and a stuff offset. It is assumed that software will initialize the TCP checksum field in the TCP frame to compensate for the fact that the hardware calculates the checksum over the actual IP header, rather than the pseudo-IP header. The 16-bit TCP checksum is computed on-the-fly during frame data transfer between the PCI Bus Interface and the TX FIFO.

Tag bits generation

As the frame is being written into the TX FIFO, end of the frame is denoted by tag bits in two consecutive words: the last data word and the control word.

2.4.3 RX DMA block

The RX DMA block works with the PCI BIF block to move receive frames into the host memory. The RX DMA block handles TCP checksum generation. The RX DMA attempts to move data on cache line boundaries between the RX FIFO and the PCI BIF block. Handling of misaligned bursts due to PCI retries is handled within the PCI BIF and is transparent to the TX DMA.

2.4.4 TX and RX DMA FIFOs

The TX and RX FIFOs are implemented as RAM arrays, where both sides of each FIFO are entirely under their respective DMA block control. Given the role these FIFOs play in GEM's functionality, along with the size and speed demands placed on these blocks, it is important to provide them with flexible diagnostics mechanisms.

Every FIFO location is accessible using PIO instructions. The access is based on a loadable pointer register, and a set of Data Registers. The FIFO read / write granularity is in 65 bits entries (64 data bits plus the tag bit).

2.4.5 MAC blocks

The Ethernet MAC core implements the IEEE 802.3 MAC protocol for CSMA/CD networks, and 802.3x for the implementation of the MAC Control Sublayer for Full Duplex with Flow Control operation. These standards presently specify operation up to speeds of 100Mbps. GEM operation above 100Mbps is defined to comply with the 802.3z standard in development.

Figure 2-9 depicts the block diagram of the MAC block.

2.4.5.1 Host Interface Buffer (HIB)

- Implements the PIO interface between the BIF and the MAC core

2.4.5.2 Transmit MAC (TxMAC)

- Implements the IEEE 802.3 transmit portion of the protocol.
- Implements the slave interface handshake between the TX FIFO and the TxMAC for frame transfers.
- Performs the synchronization between the local clock domain and the transmit media clock domain in the transmit data path.

2.4.5.3 Receive MAC (RxMAC)

- Implements the IEEE 802.3 receive portion of the protocol.
- Implements the slave interface handshake between the RX FIFO and the RxMAC for frame transfers.
- Performs the synchronization between the local clock domain and the receive media clock domain in the receive data path.

2.4.5.4 Transceiver Interface (XIF)

- Implements the MII interface protocol (excluding the Management portion).
- Performs the nibble-to-byte and byte-to-nibble conversion between the protocol engine and the MII.

2.4.5.5 Control MAC (CMAC)

- Generates MAC Control Frames using pre-programmed fields.
- Enforces hysteresis rules for PAUSE XON/XOFF MAC Control Frames before emission.
- Detects MAC Control Frames and loads the PAUSE Timer appropriately.

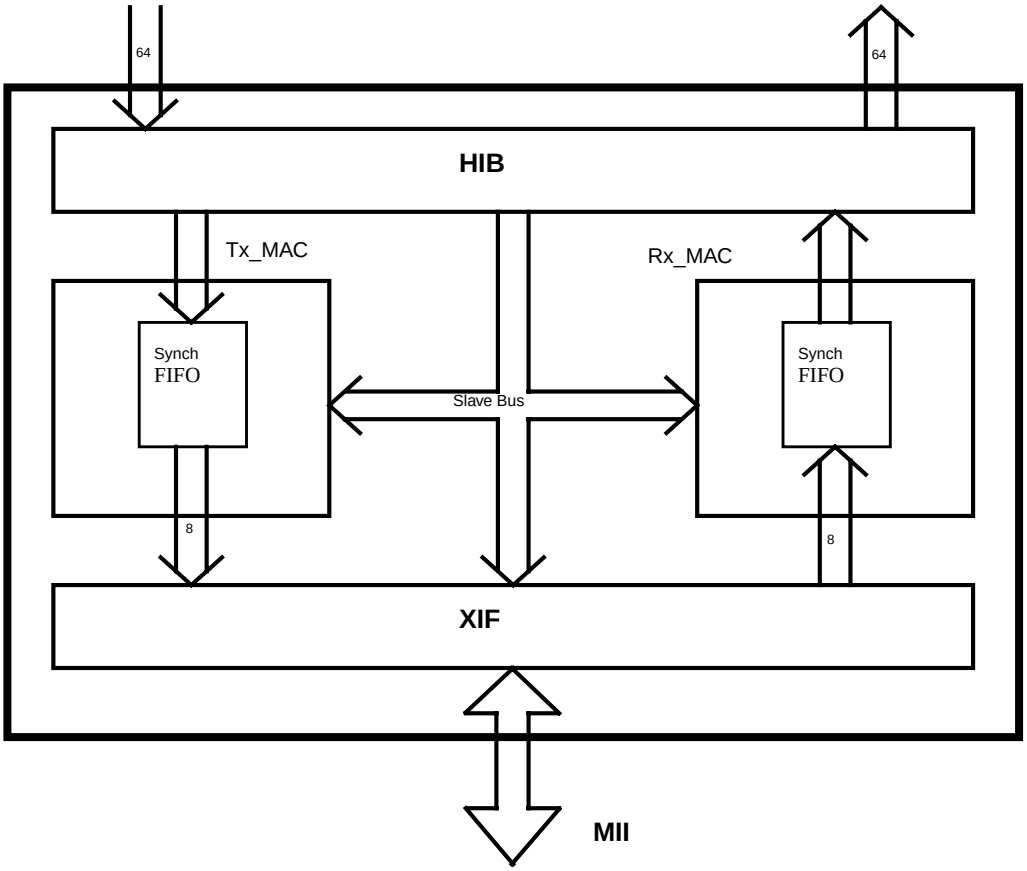


Figure 2-9 MAC block

2.4.5.6 *Transceiver Management Interface (MIF) block*

The Management Interface (MIF) implements the management portion of the MII protocol. It allows the host to program and collect status information from the MII transceiver. The MIF supports three modes of operation:

"Bit-Bang" Mode

This mode of operation provides maximum flexibility with minimum hardware support for the serial communication protocol between the host and the transceivers. The actual protocol is implemented in software, and the interaction with the hardware is done via three one-bit registers: data, clock and output_enable. Each read / write operation on a transceiver register would require approximately 150 software instructions by the host.

"Frame" Mode

This mode of operation provides a much more efficient way of communication between the host and the transceivers. The serial communication protocol between the host and the transceivers is implemented in hardware, and the interaction with the software is done via one 32-bit register (Frame Register). When the software wants to execute a read/write operation on a transceiver register, all it has to do is load the Frame Register with a valid instruction ("frame"), and poll the Valid Bit for completion. The hardware will detect the instruction, serialize the data, execute the serial protocol on the MII Management Interface and set the Valid Bit to the software upon completion.

Polling Mode

As defined in 802.3u, a transceiver shall implement at least one status register that will contain a defined set of essential information needed for basic network management. Since the MII does not include an interrupt line, a polling mechanism is required for detecting a status change in the transceiver. In order to reduce the software overhead, the above mentioned polling mechanism has been implemented in hardware. When this mode of operation is enabled, the MIF will continuously poll one specified register in each transceiver in a round-robin fashion, and generate a maskable interrupt when a status change is detected. Upon detection of an interrupt, the software can read a Local Status Register that will provide the latest contents of the transceiver register, and an indication which bits have changed since it was last read. This mode of operation can only be used when the MIF utilizes the "Frame Mode" for normal communication with the transceivers.

2.4.6 Hash Filtering Mechanism for Multicast Addresses

The MAC implements a hash filtering function to aid software in determining whether or not a packet with a multicast destination addresses should be accepted. The host manages a list of multicast addresses that it will respond to. For each address in the list the driver uses the hash routine to set the appropriate bit in one of the sixteen 16-bit Hash Table registers in the MAC. A device can be made a member of several groups by setting the appropriate bits in the Hash Table. In the MAC if a packet is received with the multicast bit set in the destination address, the incoming address is sent through the 802.3 32-bit CRC function. This circuit is the hash function. The high order 8 bits of the resultant 32-bit CRC is used in reverse order to select one of the 256 bit positions in the Hash Table. If the selected bit in the table is set to '1', the packet will be accepted and passed up to the host. In addition, the hardware passes up a Hash Pass bit and a Hash Value in that packet's descriptor. When the packet is received by the host and the Hash Pass bit is set, it must search the multicast address list to see if the packet's address is in the list. If it is not in the list, the packet is discarded. The Hash Value is the most significant 16 bits of the resultant 32-bit CRC value. It can aid the host in implementing a software based Hash Table of up to 64k entries.

2.4.7 PCS 8B/10B Block

The 8B / 10B implements the coding function, equivalent to Fiber Channel FC-1. The 10B code output of this block consists of 10-bit DC balanced Transmission Characters that reflect the state of the TxMAC and the byte stream generated by it. Special transmission characters, called Ordered sets, are generated by this block to convey frame boundary information, as well as maintaining the link during idle periods.

Similarly, this block relies on the 10B codes arriving from the far end of the link, and derives from them word alignment, frame boundary as well as the actual receive frame data bytes to be passed to the RxMAC.

This block's complies with the Fiber Channel 10-bit interface draft X3 Technical Report, allowing GEM to be used with commercially available SERDES chips via its 10-bit interface pins.

2.4.7.1 Notation Conventions

This specification uses the FC-1 letter notation to name 10-bit Transmission Characters. Transmission Characters are named using the unencoded information byte (as presented at the GMII, for example).

Transmission Characters are represented as either K**xx.y** or D**xx.y**, where K is used for Special Codes and D is used for valid Data bytes. The **xx** field corresponds to the decimal value of the five least significant bits of the information byte, while **y** corresponds to the value of the three most significant bit. For example a preamble pattern consists of the data byte 0x55, which in FC-1 letter notation would be D21.2 (five lsb = 10101 = 21 decimal, and three msb=010 is the suffix 2).

Note that the letter notation does not define the 10-bit encoding itself. The 10-bit encoding is given in tabular form and includes one version of the character encoding for positive and one version for negative Running Disparity (both versions are identical for neutral encodings).

2.4.7.2 8B/10B Codes

The Valid Data Character codes are identical to those defined by FC-1, Table 22, in FC-PH Rev 4.3.

Valid Special Characters are also defined as per FC-1. Their codes are:

Table 2-3 Valid Special Characters

Special Code	Current RD -	Current RD+
	abcdei fghj	abcdei fghj
K28.0	001111 0100	110000 1011
K28.1	001111 1001	110000 0110
K28.2	001111 0101	110000 1010

Table 2-3 Valid Special Characters

K28.3	001111 0011	110000 1100
K28.4	001111 0010	110000 1101
K28.5	001111 1010	110000 0101
K28.6	001111 0110	110000 1001
K28.7	001111 1000	110000 0111
K23.7	111010 1000	000101 0111
K27.7	110110 1000	001001 0111
K29.7	101110 1000	010001 0111
K30.7	011110 1000	100001 0111

Ordered Sets are combinations of Special Characters and Data Characters used for link initialization, and frame delineation. Ordered Sets can be one, two or four characters long. The Ordered sets used by GEM is specific to 802.3z and is currently defined as:

Table 2-4 802.3z Ordered Sets

Code	Function	Ordered Set
C1	Link Configuration 1	K28.5 D21.5 **configuration**
C2	Link Configuration 2	K28.5 D2.2 **configuration**
I1	Idle/Flip Disparity	K28.5 D5.6
I2	Idle/no flip	K28.5 D16.2
S	SOP	K27.7
T	EOP1	K29.7
R	EOP2	K23.7
H	EOP invalid	K30.7

Where **configuration** carries a 16 bit value exchanged during the Auto-negotiation phase (normally identical to the PCS MII Advertisement Register).

2.4.7.3 PCS link initialization and Auto-negotiation

The PCS function is also responsible for link initialization, including Auto-negotiation. The initialization and Auto-negotiation functions apply to SERDES and Serialink interface modes. Whenever the interface mode is MII, PCS Auto-negotiation is also attempted by the PCS over Serialink. This enables the software driver to implement a *link status* based switch-over for GEM applications with MII and Serialink interfaces. The interface that has a cable connected and link UP is automatically selected.

2.4.8 Serial Link Block

This block is a full duplex transceiver, equivalent to the commonly designated SERDES (SERializer DESerializer) devices and implements a subset of the FC-0 functions. It is responsible for serializing the 10-bit codes into an NRZ serial bit stream driven by the PECL differential output. For receive, the block performs clock / data recovery on the NRZ differential input bit stream as received from the network, aligns and de-serializes the data to be presented as a 10-bit stream to the 8B / 10B block.

2.4.8.1 Serial Link Block functions

- Full Duplex physical layer interface at 1.25 Gbps serial transfer rate with BER of less than 10^{-12} .

- 10 bit parallel data paths to PCS.

- Can be powered down via software control when the block is not used.

- Serial and parallel loopback for diagnostics.

- 50 and 75 ohm line impedances with external termination resistors.

2.4.9 Diagnostic LEDs

LEDs intended to serve as installation aid comprise:

- LINK OK
- FULL DUPLEX
- TRANSMIT ACTIVITY
- RECEIVE ACTIVITY
- COLLISION

Link OK and Full Duplex conditions are reported by the PCS and only apply to configurations that use GEM's 8B / 10B PCS function. For MII / GMII modes the software driver may activate the Link and Full Duplex LEDs using the XIF Register. It is also possible to rely on the MII transceiver LEDs for these modes.

The remaining LEDs are logically derived from the MII control signals and apply to MII / GMII, SERDES and Serialink modes.

The Transmit LED is derived from TXEN active. The Receive LED from RXDV, and the Collision LEDs from the COL signal.

All LED signals except LINK OK and FULL DUPLEX are pulse stretched to at least 100msec duration.

2.5 Interfaces and Data Paths

2.5.1 PCI Interface

Basic Feature Set

The major features of the PCI interface are:

- PCI interface compliant with PCI spec revision 2.1
- 64 bit data path master interface
- 66.6 MHz bus clock operation. Bus interface will run at lower frequencies up to 66.6 MHz (e.g. 33.3 MHz) with lower bus interface performance.
- Supports 32 bit bus when installed in a 32 bit bus slot
- 32 bit data path slave interface.

Universal IO

Operates in 5V and 3.3V slots.

2.5.2 MII/GMII Interface

As an MII this interface can be used with 100BASE-T transceivers. The implementation of the interface conforms with the MII specification. The MII data path is defined as two uni-directional nibble-wide buses: one for transmit and one for receive, with transmit and receive clocks supplied by the transceiver. The MII is clocked at 25MHz or less, and may be connectorized.

GMII is an extension of the MII that retains the logical functionality of its control signals, while it widens the data path to two uni-directional byte-wide buses, and it adds support for *carrier extension* characters. In addition to the transceiver supplied clocks, the MAC provides a transmit clock output that may be used by the transceiver to sample transmit data and control signals with less stringent clock skew requirements. The GMII is clocked at 125 MHz or less, and is an inter chip interface with no intervening connectors.

The MII/GMII management interface utilizes a 1-bit wide bi-directional serial line.

2.5.3 Expansion ROM Interface

A byte-wide PROM may be used as a GEM's expansion ROM. This device would typically store the Fcode image as well as the unique MAC address for the board. The maximum PROM size is 64kbytes. The interface only supports PROM reads. The MAC address is written into the PROM at ICT time as part of the PCI expansion ROM VPD structure.

2.5.4 PIO Bus Interface

The PIO Strategy is described below.

1) All PIO accesses are 32 bits wide, with the following exceptions.

- Configuration space registers supports 8, 16 and 32 bit accesses.

- On board PROM supports 8, 16 and 32 bit accesses.

2) All blocks receive a dedicated chip select signal.

3) All blocks return a dedicated ready signal, except TxMAC and RxMAC.

4) All blocks share the same 32 bit bus for PIO writes.

5) All blocks return PIO read data on a point to point, dedicated PIO read data bus.

6) TxMAC and RxMAC return read data within a bounded number of clocks.

7) Write data for TxMAC and RxMAC will be kept stable for a bounded number of clocks.

8) PIO reads from the PROM go the through the PROM controller block. The BIF block indicates the size of access to the PROM controller block. For 16/32 bit accesses, PROM bytes are packed to complete the access.

9) All PIO writes are registered in BIF.

10) The PIO reads that cannot be completed in 16 clock cycles will be issued a retry. Except PROM accesses, all other PIO read accesses are completed within 16 clock cycles.

2.5.5 RxDMA <-> MAC Interface

This non-exposed interface is utilized for transferring incoming packet data from the synchronization fifo in the RxMAC to the receive fifo in the RxDMA. All signals are active high.

Signal Description

- **rx_data[63:0]**
  This data bus is used to transfer eight bytes of receive packet data or packet status from the RxMAC to the RxDMA. These lines shall present the data to the RxDMA in a “little endian” format.
- **rx_tag**
  This signal is used to indicate the completion of a packet transfer from the RxMAC to the RxDMA. For each receive packet, the last data double-word and the status double-word shall have this bit set. The remaining double-words of the packet shall have this bit cleared.
- **rx_mac_req**
  This signal is driven by the MAC and is used to indicate to the RxDMA that the MAC is ready to start the transfer of the next burst of data.
- **rx_ack**
  This signal is driven by the RxDMA and is used to indicated to the MAC that the RxDMA is executing a burst transfer.

- **tx_fc_xoff_req**
  This signal is driven by the RxDMA and is used to indicate to the MAC that a “Pause” Flow Control frame should be transmitted on the medium, with a pause parameter as programmed in the Pause Value Register in the CMAC.
- **tx_fc_xon_req**
  This signal is driven by the RxDMA and is used to indicate to the MAC that a “Pause” Flow Control frame should be transmitted on the medium, with a pause parameter equal to zero.
- **tx_fc_ack**
  This signal is driven by the MAC, and is used to indicated to the RxDMA that the MAC has transmitted the “Pause” Flow Control frame previously requested by the RxDMA using the tx_fc_xoff_req/tx_fc_xon_req request lines.

Protocol Description

Data from a receive packet that arrives from the medium is first stored in the synchronization fifo in the RxMAC. After enough data has been accumulated to execute a burst (32 bytes) or the end of the packet has been detected, the MAC asserts the request line to the RxDMA (rx_mac_req).

Upon detection of a request, and if the receive queue can accept the burst, the RxDMA shall assert the acknowledgment (rx_ack) **exactly one clock** after the assertion of the request. Once set, it shall remain asserted **for exactly 4 clocks**, if a full 32-byte burst transfer is executed. If the end of the packet is detected in the middle of a burst, the RxDMA **shall de-assert** the rx_ack after the transfer of the status word has been completed. In response to rx_ack, the MAC **may or may not** de-assert the request line.

Upon detection of rx_ack, the MAC will start driving the first eight bytes of valid data on the rx_data lines at the next rising edge of the local clock, and will switch to the next two consecutive bytes of the packet for three more clock edges. Thus, the acknowledgment signal performs framing of a 32-byte burst transfer between the MAC and the RxDMA.

At the end of a packet transfer, the MAC appends to it a status/control word after the last word that contains at least one valid byte of data. The “invalid” bytes in the last data word of the packet are driven to 0x00 by the MAC.

Upon detection of congestion in the receive data path, the RxDMA may assert the tx_fc_xoff_req signal to the MAC, requesting a transmission of a flow control pause frame to the remote end of the link. The pause time is expected to be programmed by the software in the Pause Value Register. The MAC will complete the transmission of the frame that is currently in progress, transmit the pause frame (when the transmit link becomes free) and acknowledge the request. In response to tx_fc_ack, the RxDMA shall de-assert the tx_fc_xoff_req signal.

Once the RxDMA detects that the receive data path is no longer congested, it may assert the tx_fc_xon_req signal to the MAC, requesting the transmission of a flow control pause frame with the pause time equal to zero. This effectively enables the transmission from the remote end of the link. The MAC will com-

plete the transmission of the frame that is currently in progress, transmit the pause frame and acknowledge the request. In response to tx_fc_ack, the RxDMA shall de-assert the tx_fc_xon_req signal.

Status Word Format

The format of the receive packet status word, generated by the MAC, is as follows:

[63]:	Abort:	Indicates to the RxDMA that the packet that is currently being transferred is not valid.
[62]:	Bad_Packet:	When set, this bit indicates that the receive packet is a bad packet based on either: a CRC mismatch detected by the RxMAC, or invalid coding reported by the PCS block.
[61]:	Alternate:	When set, this bit indicates that the packet' DA matched the alternate address (stored in MAC Address 3,4,5 Registers).
[60]:	Hash_Pass:	When set, this bit indicates that the received packet's DA has passed the hashing filter algorithm in the RxMAC.
[59:44]:	Hash_Value:	This field provides the value of the received packet's hashed DA, as computed by the RxMAC.
[43:31]:	Reserved:	Set to 0.
[30:16]:	Pkt_Length:	This field provides the length of the received packet (in bytes), as computed by the RxMAC.
[15:0]:	Reserved:	Set to 0.

2.5.6 TxDMA <-> MAC Interface

This non-exposed interface is utilized for transferring the outgoing packet data from the TX FIFO in the TxDMA to the synchronization fifo in the TxMAC. All signals are active high.

Signal Description

- **tx_data[63:0]**
  This data bus is used to transfer 8 bytes of transmit packet data or packet status from the TxDMA to the MAC. These lines shall present the data to the MAC in a “little endian” format.
- **tx_tag**
  This signal is used to indicate the completion of a packet transfer from the TxDMA to the MAC. For each transmit packet, the last data double-word and the status double-word shall have this bit set. The remaining double-words of the packet shall have this bit cleared.
- **tx_mac_req**
  This signal is driven by the MAC and is used to indicate to the TxDMA that the MAC is ready to start the next burst of data transfer.
- **tx_ack**
  This signal is driven by the TxDMA and is used to indicated to the MAC that the TxDMA is executing a 32-byte burst transfer.
- **tx_retry_req**
  This signal is driven by the MAC and is used to indicate to the TxDMA that the MAC is requesting a re-transmission of the current packet from the beginning, due to a collision on the shared medium.

Protocol Description

After the MAC completes the transmission of a packet on the medium, it asserts the tx_mac_req line to the TxDMA. If at least one packet is ready to be transmitted, the TxDMA asserts the tx_ack **exactly one clock** after the assertion of the request. Once set, it shall remain asserted **for exactly 4 clocks**, if a full 32-byte burst transfer is executed. If the end of the packet is detected in the middle of a burst, the TxDMA **shall de-assert** the tx_ack after the transfer of the control/status word has been completed. In response to tx_ack, the MAC **may or may not** de-assert the request line.

Simultaneously with the assertion of the acknowledgment signal, the TxDMA starts driving the first eight bytes of valid data on the tx_data lines. It will switch to the next eight consecutive bytes of the packet at the next rising edge of the local clock, and will repeat this action for two more consecutive clock edges. Thus, the acknowledgment signal performs “framing” of a 32-byte burst transfer between the TxDMA and the MAC.

At the end of a packet transfer, the TxDMA appends to it a status/control entry after the last double-word that contains at least one valid byte of data.

If during a packet transfer the TxMAC encounters a collision on the medium, it asserts the tx_retry_req signal together with the tx_mac_req. In response, the TxDMA prepares itself for packet re-transmission by rewinding its Read Pointer to the beginning of the current packet plus 32 bytes, and asserts the tx_ack signal **within a maximum of 4 clock cycles** after the assertion of tx_mac_req.

The MAC may assert the tx_retry_req signal at any time during the transfer of the data portion of the packet, or after the entire packet has been transferred to the MAC. The TxDMA must monitor the tx_retry_req signal of the first burst of a packet transfer to the MAC in order to determine whether the previous packet has been successfully transmitted or needs to be re-transmitted.

Control Word Format

The format of the transmit packet control word, generated by the OPP, is as follows:

[63]:	Abort:	Indicates to the TxMAC that the packet that is currently being transferred for transmission is invalid. The Tx_MAC will generate a TX_ER indication on the MII. This bit is always set to zero in GEM.
[62]:	No_CRC:	Indicates to the TxMAC not to generate a CRC field for the current packet. In GEM this bit corresponds to the “No CRC” bit in the TX Descriptor.
[61:59]:	LBB:	Last Byte Boundary. This field indicates to the TxMAC the position of the last valid byte of data in the previously transferred last data word.
[58: 0]:	Reserved:	This field is ignored by the TxMAC.

2.6 Error Conditions and Recovery

There are two types of error conditions that can be encountered during the normal operation: fatal errors and non-fatal errors.

Fatal errors are errors that should never occur. They usually indicate a serious failure of the hardware or a serious programming error. When this type of error occurs, the recovery process is non-graceful and involves software reset after the appropriate actions were taken to correct the failure. Fatal error events are always reported to the software via interrupts.

Non-fatal errors are errors are that are expected to occur when certain conditions occur on the network or in the system. When this type of error occurs, a graceful recovery mechanism is provided via a combination of hardware and software as described below. Non-fatal errors may or may not be reported to the software.

2.6.1 Non-Fatal Errors

Rx buffer not available

This condition is encountered when no more receive descriptors are available for receive frames. An interrupt is generated to the device driver to indicate the occurrence of this event. The RX DMA clears this condition as a result of the host CPU updating the RX Kick Register after posting new buffers.

Due to flow control, and the RX FIFO size, this condition does not necessarily mean that receive data is being discarded. Data discarding is reported by the *Rx FIFO overflow* condition.

Rx buffer overflow

The RX DMA transfers frames from the buffer memory to the host memory. GEM will not detect if the size of a buffer available in the host memory is smaller than the frame size, therefore software is responsible for allocating buffers large enough to fit the value programmed in the *Maximum Frame Size Register* in the MAC. Frames that exceed this value are truncated by the MAC, and the buffer re-claimed using the Rx Abort mechanism described below. Software can determine if such longer frames exist on the network and are directed to GEM by reading the *Length Error Counter.*

Rx Abort from the MAC

A receive frame may be aborted while in transit from the network to host memory. Depending on how far into the frame the abort condition is detected, different blocks might be involved in the recovery.

In some cases the condition is detected early on by the RX MAC, even before the frame was presented to the RX FIFO interface. In this case the frame is silently dropped by the RX MAC with no RX DMA involvement.

If the abort condition is detected after transfer to the RX FIFO has begun, it is signalled by setting the “abort” bit in the status word. If the RX DMA observes the “abort” bit set before it has started removing the frame from the RX FIFO,

the frame will be discarded by rewinding the RX FIFO Write Pointer back to the value of the Write Shadow Pointer. Otherwise, when it is too late to rewind the Write Pointer and the RX DMA is responsible for discarding the frame and re-using its descriptor.

RX Abort conditions are not reported as such to the host, but the events causing it (BAD CRC, fragment, RX_ER, RX FIFO Overflow, etc.) have their own reporting mechanisms.

Bad frame reception, for troubleshooting and network monitoring purposes, may be enabled by setting the Err_Check_Disable bit in the *RX_MAC Configuration Register*. In this case the “abort” bit in the status word is never set. All bad frames are received, including fragments shorter than the value programmed into the *Minimum Frame Size Register*.

Rx FIFO overflow

If the RX FIFO becomes unable to receive more data from the RX MAC, in spite of the Flow Control mechanism, this condition propagates to the RX MAC. The RX MAC will discard data when it runs out of space in its synchronization FIFO, and optionally generate an interrupt by setting the Rx_Overflow bit in the *RxMAC Status Register*. Partial frames resulting from this condition are discarded by setting the “abort” bit in the status word.

2.6.2 PCI Error handling

GEM’s PCI Error handling follows section 3.8 of the PCI Spec revision 2.1, and allows recovery at the device driver level, whenever possible, by using the PCI Error Status Register when fatal errors occur.

PCI Bus Errors, like parity errors, may result in fatal or non fatal errors, depending on the nature of the error and the programming of PCI Configuration space bits. The bits involved are:

Parity Error Response - PCI Command Register, bit 6

SERR# Enable - PCI Command Register, bit 8

Detected Parity Error - PCI Status Register, bit 15

Signaled System Error - PCI Status Register bit, 14

Data Parity Error Detected - PCI Status Register, bit 8

Even parity generation is the responsibility of the agent that drives the PCI address/data bus for any given phase. GEM’s PCI Bus parity error response is determined by the Parity Error Response and SERR# Enable bit values programmed in PCI Configuration Space.

2.6.2.1 PCI Slave Bus Parity Errors

Slave parity errors are non-fatal and are reported using the Set Detected Parity Error bit in PCI Configuration space. Table 2-5 defines GEM’s response to slave access parity errors.

Table 2-5 Parity Errors on Slave GEM Accesses

Parity Error During	**PARITY ERROR RESPONSE = 0**	**PARITY ERROR RESPONSE = 1**
Address Phase or Special Cycle	Set Detected Parity Error bit, if cycle is owned, claim and terminate normally	Set Detected Parity Error bit, do not claim the cycle. If SERR# Enable bit is set assert SERR# and set Signaled System Error bit.
Data Phase (READ)	Not applicable	Not applicable
Data Phase (WRITE)	Set Detected Parity Error bit, terminate cycle normally	Set Detected Parity Error bit, assert PERR#, terminate cycle but ignore data.

2.6.2.2 *PCI Master Bus Parity Errors*

Master data phase parity errors (occurring while GEM DMA owns the PCI bus) are treated as fatal when PARITY ERROR RESPONSE bit is set. Otherwise data phase errors are not fatal. Address phase parity error response depends on the response of the device detecting the error. If reported via SERR# these errors are treated by GEM as fatal. Table 2-6 defines GEM's response to master parity errors.

Table 2-6 Parity Errors on Master GEM cycles

Parity Error During	**PARITY ERROR RESPONSE = 0**	**PARITY ERROR RESPONSE = 1**
Address Phase or Special Cycle	GEM does not generate special cycles. Parity errors on GEM's DMA address phases may be reported by other agents, by activating SERR#. In this case GEM stops DMA activity and interrupts CPU. Fatal Error.	
Data Phase (READ)	Set Detected Parity Error bit, and continue DMA normally	Set Detected Parity Error and Data Parity Error Detected bits. Assert PERR#, stop DMA activity and interrupt CPU. Fatal Error.
Data Phase (WRITE)	Set Detected Parity Error bit, and continue DMA normally	Set Detected Parity Error and Data Parity Error Detected bits. Stop DMA activity and interrupt CPU. Fatal Error.

Programmer's Model 3

3.1 Address Map

3.1.1 PCI Configuration Space

GEM's PCI Configuration space follows the format of a single-function device of the PCI Bus Specification Revision 2.1. GEM's PCI Configuration space is a read / write space accessible as bytes, half-word and word, unless indicated otherwise.

GEM's PCI Configuration space is accessible at all times. Values programmed into PCI Configuration space will not be affected by software reset, and only return to their default values upon hardware reset.

Configuration cycles for unimplemented registers in GEM's PCI Configuration space are always completed normally, with Read cycles returning all zeros, and write cycles discarding the write data.

Table 3-1 GEM' PCI Configuration Space

Offset	Size	R/W	Name
0x000 - 0x001	16 bit	RO	Vendor ID=0x108E
0x002 - 0x003	16 bit	RO	Device ID=0x2BAD
0x004 - 0x005	16 bit	R/W	Command Register
0x006 - 0x007	16 bit	R/W	Status Register
0x008	8 bit	RO	Revision ID = 0x01
0x009-0x00B	24 bit	RO	Class = 0x020000 Network Controller, Ethernet
0x00C	8 bit	R/W	Cache Line Size, in 32 bit units, resets to 0x00 (values > 128 bytes, or non power of 2 are written as zero)
0x00D	8 bit	R/W	Latency Timer
0x00E	8 bit	RO	Header Type = 0x00, single function device, standard header type
0x00F	8 bit	RO	BIST. Resets to 0x00, Not capable.
0x010 - 0x013	32 bit	R/W	Base Address Register Bits[20:0] read as zero 2 Mbyte register space is non prefetchable, mapped in 32 bit Memory Space Resets to 0x0000
0x014 - 0x02D			Reserved - Read as zero
0x02E - 0x02F			Subsystem ID - value loaded from the board through pins P_D[7:0] upon deassertion of RST#.
0x030 - 0x033	32 bit	R/W	Expansion ROM Base Address, Bits [19:1] read as zero. 1 Mbyte space mapped in 32 bit Memory Space Resets to 0x0000
0x034 -0x03B			Reserved - Read as zero
0x03C	8 bit	R/W	Interrupt Line
0x03D	8 bit	RO	Interrupt Pin = 0x01 => uses pin INTA#
0x03E	8 bit	RO	Min_Gnt =0x40 => 16usec bursts
0x03F	8 bit	RO	Max_Lat=0x40 => 16usec latency
0x040-0x0FF			Reserved, read as zero

Note – The Cache Line Register supports cache line values of up to 128 bytes. For cache line sizes of zero, or larger than 128 bytes, GEM will not use PCI cache commands (MWI, MRL, MRM), and bursts are controlled assuming 32 bytes per cache line.

3.1.1.1 Command Register

The command register provides coarse control over a function's ability to generate and respond to PCI cycles. When a 0 is written to this register, the function is logically disconnected from the PCI bus for all accesses except configuration accesses.

Note – Please refer to §6.2.2 of the PCI bus specification for more information on the command register

Table 3-2 PCI Command Register

Bit	Usage
0	IO Space — Not implemented, read back as zero
1	Memory Space — Controls a function's response to memory space accesses: when set, allows the function to respond to memory space accesses. Reset to zero.
2	Bus Master — Controls a function's ability to act as a master on the PCI bus: when set, allows the device to behave as a bus master. Reset to zero
3	Special Cycles — Not implemented, read back as zero
4	Memory Write and Invalidate Enable — Controls whether a master can generate the Memory Write and Invalidate command (when set.) Reset to zero.
5	VGA Palette Snoop — Not implemented, read back as zero
6	Parity Error Response — This bit controls the function's response to parity errors. Parity errors are ignored when this bit is zero, and reported when this bit is one. Parity must be generated regardless of the value of this bit. Resets to zero.
7	Wait Cycle Control — Address/Data Stepping not implemented, read back as zero
8	SERR# Enable — This bit is an enable for the SERR# driver: when set, the SERR# pin driver is enabled. Resets to zero.
9	Fast Back-to-Back Enable — Not implemented, read back as zero
15 - 10	Reserved, read back as zero

3.1.1.2 Status Register

The status register is used to record information for PCI bus related events. Reads to this register behave normally; during writes, bits can only be reset, but not set. A bit is reset whenever the register is written, and the data in the corresponding bit location is a 1.

Note – Please refer to §6.2.3 of the PCI bus specification for more information on the status register

Table 3-3 PCI Status Register

Bit	Usage
0-4	Reserved, read back as zero
5	66MHz Capable — Read-only, set to 1, indicates GEM is capable of running at 66MHz
6	UDF — Read-only, set to 0, indicates no User Definable Features
7	Fast Back-to-Back Capable — Read-only, set to 1, indicates GEM as a target is capable of accepting fast back-to-back transactions with different agents
8	Data Parity Error Detected — Set when three conditions are met: 1) PERR# was asserted or observed asserted, 2) GEM was the bus master for the transaction in which the error occurred, 3) Parity Error Response bit in Command Register is set. Resets to zero.
9-10	DEVSEL Timing — Read-only, set to 01 (medium)
11	Signaled Target Abort — When set, indicates that an illegal access to GEM was terminated by GEM with a target-abort. Resets to zero.
12	Received Target Abort — When set, indicates a GEM bus master transaction terminated with a target-abort. Resets to zero.
13	Received Master Abort — When set, indicates a GEM bus master transaction terminated with a master-abort. Resets to zero.
14	Signaled System Error — Set when GEM asserts SERR#.
15	Detected Parity Error — Set when GEM detects a parity error, even if the Parity Error Response in the Command Register is zero. Resets to zero.

3.1.2 Expansion ROM space

The Expansion ROM space is a 1 Mbyte read only memory space through which byte wide external PROMs of up to 64 kbytes may be read during POST. This space is accessible as 8, 16, or 32-bit reads. The system maps and enables this space using the Expansion ROM Base Address Register in PCI Configuration Space.

3.1.3 GEM Register Space

This is a 2Mbyte space through which internal registers as well as the external PROM may be accessed.

This space is mapped in 32 bit memory space, and is accessible as 32-bit words. Register addresses are offset from the value written in the Base Address Register in PCI Configuration Space.

GEM's registers are not accessible in PCI IO space.

Table 3-4 GEM Register Address Map

Address Offset	R/W	Default	Actual Size (bits)	Description
Global Resources				
0x0000	RO	0x0	3	SEB State Register
0x0004	RW	0x00000	17	Configuration Register
0x000C	R-AC	0x00000000	25	Status Register (auto-clears top level bits)
0x0010	RW	0xFFFFFFF	12	Interrupt Mask Register
0x0014	WO	0x00000000	6	Interrupt Ack Register
0x001C	RO	0x00000000	25	Status Register Alias (no auto-clear)
0x1000	R-AC	0x0	3	PCI Error Status Register
0x1004	RW	0x7	3	PCI Error Mask Register
0x1008	RW, RO bit	0x0	4	BIF Configuration Register
0x100C	RW	0x00000000	32	BIF Diagnostic Register
0x1010	RW- 2 AC bits	0x00	3	Software Reset Register
Transmit DMARegisters				
0x2000	RW	0x0000	13	TX Kick Register
0x2004	RW	0x118010	22	TX Configuration Register
0x2008	RW	undefined	21	TX Descriptor Base Low
0x200C	RW	undefined	32	TX Descriptor Base High
0x2010	RO			Reserved
0x2014	RW	0x000	11	TX FIFO Write Pointer
0x2018	RW	undefined	11	TX FIFO Shadow Write Pointer
0x201C	RW	0x000	11	TX FIFO Read Pointer
0x2020	RW	undefined	11	TX FIFO Shadow Read Pointer
0x2024	RO	0x000	11	TX FIFO Packet Counter
0x2028	RO	0x00000000	22	TX State Machine Register
0x202C				
0x2030	RO	0x00000000	32	TX Data Pointer Low
0x2034	RO	0x00000000	32	TX Data Pointer High
0x2100	RO	0x0000	13	TX Completion Register
0x2104	RW	undefined	11	TX FIFO Address
0x2108	RO	undefined	1	TX FIFO Tag

Table 3-4 GEM Register Address Map

Address Offset	R/W	Default	Actual Size (bits)	Description
0x210C	RW	undefined	32	TX FIFO Data Low
0x2110	RW	undefined	32	TX FIFO Data HighT1
0x2114	RW	undefined	32	TX FIFO Data HighT0
0x2118	RO	0x090	11	TX FIFO Size
Receive DMA Registers				
0x4000	RW	0x1000010	27	RX Configuration Register
0x4004	RW	undefined	21	RX Descriptor Base Low
0x4008	RW	undefined	32	RX Descriptor Base High
0x400C	RW	0x000	11	RX FIFO Write Pointer
0x4010	RW	undefined	11	RX FIFO Shadow Write Pointer
0x4014	RW	0x000	11	RX FIFO Read Pointer
0x4018	RO	0x000	12	RX FIFO Packet Counter
0x401C	RO	0x00000	18	RX State Machine Register
0x4020	RW	0x000F8	18	Pause Thresholds
0x4024	RO	Undefined	32	RX Data Pointer Low
0x4028	RO	Undefined	32	RX Data Pointer High
0x4100	RW	0x0000	13	RX Kick Register
0x4104	RO	0x0000	13	RX Completion Register
0x4108	RW	0x00000	17	RX Blanking Register
0x410C	RW	undefined	12	RX FIFO Address
0x4110	RO	undefined	1	RX FIFO Tag
0x4114	RW	undefined	32	RX FIFO Data Low
0x4118	RW	undefined	32	RX FIFO Data HighT0
0x411C	RW	undefined	32	RX FIFO Data HighT1
0x4120	RO	0x140	11	RX FIFO Size
MAC Registers				
0x6000	RW	0x0	1	TX MAC Software Reset Command
0x6004	RW	0x0	1	RX MAC Software Reset Command
0x6008	RW	0x0XXXX	17	Send Pause Command Register
0x6010	R-AC	0x000	9	TX MAC Status Register

Table 3-4 GEM Register Address Map

Address Offset	R/W	Default	Actual Size (bits)	Description
0x6014	R-AC	0x00	7	RX MAC Status Register
0x6018	R-AC	0xXXXX0000	32	MAC Control Status Register
0x6020	RW	0x1FF	9	TX MAC Mask Register
0x6024	RW	0x7F	7	RX MAC Mask Register
0x6028	RW	0x7	3	MAC Control Mask Register
0x6030	RW	0x000	9	TX MAC Configuration Register
0x6034	RW	0x00	8	RX MAC Configuration Register
0x6038	RW	0x0	3	MAC Control Configuration Register
0x603C	RW	0x00	7	XIF Configuration Register
0x6040	RW	undefined	8	InterPacketGap0 Register
0x6044	RW	undefined	8	InterPacketGap1 Register
0x6048	RW	undefined	8	InterPacketGap2 Register
0x604C	RW	undefined	10	SlotTime Register
0x6050	RW	undefined	10	MinFrameSizeRegister
0x6054	RW	undefined	30	MaxFrameSize Register
0x6058	RW	undefined	10	PA Size Register
0x605C	RW	undefined	4	JamSize Register
0x6060	RW	undefined	8	Attempt Limit Register
0x6064	RW	undefined	16	MAC Control Type Register
0x6080	RW	undefined	16	MAC Address 0 Register
0x6084	RW	undefined	16	MAC Address 1 Register
0x6088	RW	undefined	16	MAC Address 2 Register
0x608C	RW	undefined	16	MAC Address 3 Register
0x6090	RW	undefined	16	MAC Address 4 Register
0x6094	RW	undefined	16	MAC Address 5 Register
0x6098	RW	undefined	16	MAC Address 6 Register
0x609C	RW	undefined	16	MAC Address 7 Register
0x60A0	RW	undefined	16	MAC Address 8 Register
0x60A4	RW		16	Address Filter 0 Register
0x60A8	RW		16	Address Filter 1 Register
0x60AC	RW		16	Address Filter 2 Register
0x60B0	RW		8	Address Filter 2&1 Mask Register
0x60B4	RW		16	Address Filter 0 Mask Register
0x60C0	RW	undefined	16	Hash Table 0 Register
0x60C4	RW	undefined	16	Hash Table 1 Register

Table 3-4 GEM Register Address Map

Address Offset	R/W	Default	Actual Size (bits)	Description
0x60C8	RW	undefined	16	Hash Table 2 Register
0x60CC	RW	undefined	16	Hash Table 3 Register
0x60D0	RW	undefined	16	Hash Table 4 Register
0x60D4	RW	undefined	16	Hash Table 5 Register
0x60D8	RW	undefined	16	Hash Table 6 Register
0x60DC	RW	undefined	16	Hash Table 7 Register
0x60E0	RW	undefined	16	Hash Table 8 Register
0x60E4	RW	undefined	16	Hash Table 9 Register
0x60E8	RW	undefined	16	Hash Table 10 Register
0x60EC	RW	undefined	16	Hash Table 11 Register
0x60F0	RW	undefined	16	Hash Table 12 Register
0x60F4	RW	undefined	16	Hash Table 13 Register
0x60F8	RW	undefined	16	Hash Table 14 Register
0x60FC	RW	undefined	16	Hash Table 15 Register
0x6100	RW	undefined	16	Normal Collision Counter
0x6104	RW	undefined	16	First Attempt Successful Collision Counter
0x6108	RW	undefined	16	Excessive Collision Counter
0x610C	RW	undefined	16	Late Collision Counter
0x6110	RW	undefined	16	Defer Timer
0x6114	R-AC	undefined	8	Peak Attempts Register
0x6118	RW	undefined	16	Receive Frame Counter
0x611C	RW	undefined	16	Length Error Counter
0x6120	RW	undefined	16	Alignment Error Counter
0x6124	RW	undefined	16	FCS Error Counter
0x6128	RW	undefined	16	RX code Violation Error Counter
0x6130	RW	0x000	10	Random Number Seed Register
0x6134	RO	0x00	8	State Machine Register
MIF Registers				
0x6200	RW	0x0	1	MIF Bit-Bang Clock
0x6204	RW	0x0	1	MIF Bit-Bang Data
0x6208	RW	0x0	1	MIF Bit-Bang Output Enable
0x620C	RW		32	MIF Frame/Output Register
0x6210	RW		15	MIF Configuration Register

Table 3-4 GEM Register Address Map

Address Offset	R/W	Default	Actual Size (bits)	Description
0x6214	RW	0xFFFF	16	MIF Mask Register
0x6218	R-AC		32	MIF Status Register
0x621C	RO		7	MIF State Machine Register
PCS / Serialink				
0x9000	RW	0x1000	16	PCS MII Control Register
0x9004	RO	0x0608	16	PCS MII Status Register
0x9008	RW	0x01e0	16	PCS MII Advertisement Register
0x900C	RW	undefined	16	PCS MII Link Partner Ability Register
0x9010	RW	0x8	6	PCS Configuration Register
0x9014	RW		17	PCS State Machine Register
0x9018	R-AC	0x0	1	PCS Interrupt Status Register
0x9050	RW	0xX	4	Datapath Mode Register
0x9054	RW	0x000	18	Serialink Control Register
0x9058	RW	0x0	2	Shared Output Select Register
0x905C	RO	0x0	2	Serialink State Register
PROM image				
0x100000-0x1FFFFF	RO			Expansion ROM run time access

3.1.4 Register Description

3.1.4.1 Global Resources

SEB State Register (RO)

This register reflects the internal state of the arbitration between GEM's TX and RX DMA channels. It is intended for diagnostics use only.

Table 3-5 SEB State Register

Bits	Field Name	Description
[1:0]	Arb_State	State of Arbiter
[2]	Rx_Won	Receive won internal arbitration.

Default: 0x0

Configuration Register (RW)

This register is used to configure parameters that define the DMA burst and internal arbitration behavior.

Table 3-6 Configuration Register

Bits	Field Name	Description
[0]	Infinite_Burst	When set, a DMA burst may continue for as long as it needs to transfer a packet. When clear DMA operations are limited to cache line bursts.
[5:1]	TX_DMA_Limit	Determines the number of DMA transfers (in 64 byte multiples) the TX channel may be internally granted before setting the priority to RX.
[10:6]	RX_DMA_Limit	Determines the number of DMA data transfers (in 64 byte multiples) the RX channel may be internally granted before setting the priority to TX.

Note – A change of internal arbitration priority takes effect when the present transfer is complete (namely, cache line boundaries). The default value of DMA_Limit=1 results in priority changes after 64 bytes, ensuring that priority is not switched for even the longest descriptor accesses regardless of the cache line value. Programming the corresponding DMA_Limit to a value larger than the longest packet results in priority changing on packet boundaries.

Default: 0x00042

Interrupt Status Register (R-AC)

This is the top level register used to communicate to the software events that were detected by the hardware. If a status bit is set to '1', it indicates that the corresponding event has occurred. Top level interrupts (stored in bits 0 through 6 of the register) are automatically cleared to '0' when the Status Register is read. Second level interrupts (reported in bits 13 through 18 of the

register) are cleared at the source (for example, by reading the corresponding status register). The value of the TX Completion Register is replicated in bits 19 through 31 of this register.

Table 3-7 Interrupt Status Register

Bits	Field Name	Description
[0]	TX_INT_ME	This interrupt is set when a frame with the INT ME descriptor bit set is completely transferred from the host queue to the TX FIFO.
[1]	TX_ALL	This interrupt is set when all transmit frames were transferred into the TX FIFO. Namely, the TX Kick Register and TX Completion Register values coincide. By setting the PACED_MODE it is also possible to restrict this interrupt to all transmit frames transferred AND the TX FIFO being empty.
[2]	TX_DONE	Set when any frame is transferred into the TX FIFO.
[3]	reserved	
[4]	RX_DONE	At least one frame was transferred from the RX FIFO to host memory. Set when the RX Completion Register is updated. May be delayed by the receive interrupt blanking.
[5]	Rx_Buffer_Not_Available	Set when there are no free receive buffers. Namely, the RX Kick Register and RX Completion Register values coincide.
[6]	RX_TAG_ERROR	Set when the RX FIFO tag framing is corrupted, namely anything other than two consecutive tag bits set.
[7]		
[8]		
[9]		
[10]		
[11]		
[12]		
[13]	PCS_INT	The PCS Interrupt Status Register is indicating an interrupt condition.
[14]	TX _MAC_INT	The Status Register in the TX MAC has at least one unmasked interrupt set.
[15]	RX_MAC_INT	The Status Register in the RX MAC has at least one unmasked interrupt set.
[16]	MAC_CTRL_INT	The Status Register in the MAC Control has at least one unmasked interrupt set.
[17]	MIF_Interrupt	The Status Register in the MIF has at least one unmasked interrupt set.
[18]	PCI_ERROR_INT	The PCI Error Status Register in the BIF has at least one unmasked interrupt set.
[31:19]	TX Completion	Same value as the TX Completion Register

Interrupt Mask Register (RW)

This register is used to determine which status events will cause an interrupt. If a mask bit is cleared to '0', the corresponding event causes an interrupt signal to be generated. The layout of this register corresponds bit-by-bit to the layout of the interrupt bits in the Status Register

Default: All implemented bits default high.

Interrupt Ack Register (WO)

This location may be used to clear specific top level interrupt bits. Its layout corresponds to the layout of the top level interrupt bits of the Interrupt Status Register. Bit positions written high will be cleared, while bit positions written low have no effect on the Interrupt Status Register.

Status Register Alias (RO)

This location presents the same view as the Global Interrupt Status Register, except that reading from this location does not automatically clear any of the register bits.

PCI Error Status Register (R-AC)

This register indicates PCI Errors have occurred.

Table 3-8 PCI Error Status Register

Bits	Field Name	Description
[0]	BADACK	Set if no ACK64# during ABS64 cycle.
[1]	DTRTO	Delayed transaction timeout. Set if no read retry after 2^15 clocks.
[2]	OTHER	Set by other PCI Errors. the specific error may be read from the PCI Status Register in PCI Configuration space.

PCI Error Mask Register (RW)

This register is used to determine which PCI Error status events will set the PCI_ERROR_INT in the Interrupt Status Register. If a mask bit is cleared to '0', the corresponding event causes an interrupt signal to be generated. The layout of this register corresponds bit-by-bit to the layout of the PCI Error Status Register

Default: All implemented bits default high.

BIF Configuration Register (RW except for bit 3)

This register is used to configure PCI related parameters that are not in PCI Configuration space. These values convey specific system information for the BIF block to optimize its performance. Default values indicate no special knowledge is assumed by the BIF. M66EN is a read only bit.

Table 3-9 BIF Configuration Register

Bits	Field Name	Description
[0]	SLOWCLK	SLOWCLK is used for parity error timing optimization. When PCI bus clock rates over 25MHz are used, parity results do not need to slow read acknowledge to GIO bus. IF under 25 MHz, the read acknowledge must be stalled to wait for parity results. This bit can be set to avoid slowing down the interface at typical (i.e. >= 33 MHz) speeds.
[1]	RESERVED	
[2]	B64D_DIS	When set, master won't issue 64 bit wide data cycles
[3]	M66EN	Read Only bit. Reflects the logic level of the M66EN pin. May be used by the driver to sense whether GEM is operating in a 66MHz or 33MHz PCI segment.

Default: 0x0

BIF Diagnostic Register (RW)

Table 3-10 BIF Diagnostic Register

Bits	Field Name	Description
[15:0]	RESERVED	
[22:16]	PCI Burst Controller	PCI Burst Controller state machine bits
[23]	RESERVED	
[31:24]	BIF State Machine	BIF state machine bits

Default: 0x00000000

Software Reset Register (RW-AC)

The lowest two bits of this register are used to perform an Individual Software Reset to the TX or RX functions (when the corresponding bit is set), a Global Software Reset to the GEM (when both bits are set). These bits can be set to '1'

using a Programmed IO write to the defined address. They become "self-cleared" after the corresponding reset command has been executed. The third bit(RSTOUT) is not self clearing and is used to activate the RSTOUT# pin.

Table 3-11 Software Reset Register

Bits	Field Name	Description
[0]	TX Software Reset	
[1]	RX Software Reset	
[2]	RSTOUT	When set it forces the RSTOUT# pin active (low). When clear RSTOUT# follows the level of the PCI reset input pin (RST#). This bit can be used to reset the PHY and any other circuitry connected to the RSTOUT# pin.
[31:3]	Reserved	

Programming Restrictions: *To ensure proper operation of the hardware after a Software Reset (Individual or Global), this register must be polled by the software. When both bits read back as 0's, the software is allowed to continue to program the hardware.*

3.1.4.2 TX DMA Programmable Resources

TX Kick Register (RW)

This 13-bit register is written by the host CPU with the descriptor value that follows the last valid transmit descriptor. The *TX Kick Register* and *TX Completion Register* values are used by the transmit DMA engine to control TX descriptor fetching. If the value indicates that more than one valid TX descriptor is available within the cache line being read, GEM will internally cache up to four of them. Initialized to zero on reset.

TX Completion Register (RO)

This 13-bit register stores the descriptor value that follows the last descriptor already processed by GEM. In addition to the internal use of this value for descriptor fetching, the host CPU may read this register to determine which transmit resources may be reclaimed. Descriptors up to but excluding the register value may be re-claimed. The DMA state machine updates the value of this register once the transmit data buffer contents are entirely loaded into the TX FIFO, but before setting any of the pertinent bits in the Interrupt Status Register (TX_INT_ME, TX_ALL, TX_DONE). In cases where a transmit frame has multiple descriptors this register is updated on descriptor boundaries. Initialized to zero on reset.

TX Configuration Register (RW)

This register stores parameters that control the operation of the transmit DMA channel.

Table 3-12 TX Configuration Register

Bits	Field Name	Description
[0]	Tx_DMA_Enable	When set to '1', the TX DMA channel is enabled. When cleared to '0', the TX DMA operation will orderly stop as soon as the transfer of the current data buffer has been completed. Defaults low.
[4:1]	Tx_Desc_Ring_Size	This field determines the number of descriptor entries in the Tx ring. Default: 0x8; 8k descriptor entries.
[5]	Tx_FIFO_PIO_Sel	When set to '1', the TX DMA FIFO can be accessed with PIOs using the TX FIFO address and data registers. It is recommended that the TX DMA be disabled before performing the TX FIFO PIOs in case the intent is to retain the integrity of the TX FIFO for continued transmission of packet data.
[9:6]	Reserved	
[20:10]	Reserved.	Should be set to 0x4FF.
[21]	Paced_Mode	When set to '1', the TX_ALL interrupt (bit 1 in the Status Register) will become set only after the TX FIFO becomes empty. If cleared to '0', the TX_ALL interrupt is only a function of the descriptor queue being empty. Defaults low.

Default: 0x118C10

Note – The Tx-FIFO_Threshold should be set to the maximum possible value of '4FF'. Failure to do so may create TXMAC Underrun.

Table 3-13 TX Descriptor Ring Size values

Tx_Desc_Ring_Size	Ring Size
0	32
1	64
2	128
3	256
4	512
5	1k
6	2k
7	4k
8	8k
9-15	Reserved

TX Descriptor Base Low and High (RW)

The 53 most significant bits of this register are used as the base address for the TX descriptor ring. The 11 least significant bits are not stored, and assumed to be zero.

Programming Restrictions: *The Transmit Descriptor Pointer must be initialized to a 2KByte-aligned value after power-on or software reset.*

TX FIFO Write Pointer (RW)

This 11-bit loadable counter points to the next location in the FIFO that will be loaded with TX data, the checksum or the frame control word. The counter increments upon TX FIFO loads (of 64 bits). The counter is loaded with the contents of Shadow Write Pointer when the checksum is "stuffed" into the frame. The checksum offset within the 64-bit entry is accounted for when preparing the checksum to be stuffed. This counter is used to generate the "write" address for the TxFIFO memory core.

TX FIFO Shadow Write Pointer (RW)

This 11-bit register points to the first byte of the packet that is either currently being loaded or is about to be loaded into the TX FIFO. This register serves as a temporary hold register for the Write Pointer and is used to "stuff" the checksum into the frame.

TX FIFO Read Pointer (RW)

This 11-bit loadable counter points to the next location in the FIFO that will be read from to retrieve packet data that is transferred to the TX_MAC. The counter increments on 64-bit reads from the FIFO. The counter is loaded with the contents of the Shadow Read Pointer, when a "retry" occurs due to a collision on the network. This counter is used to generate the "read" address for the TX FIFO memory core.

TX FIFO Shadow Read Pointer (RW)

This 11-bit register points to the first byte of the packet that is either currently being unloaded or is about to be unloaded from the TxFIFO. The register is loaded with the contents of the Read Pointer after the packet transfer from the FIFO to the TX_MAC has been completed. This register is used to "rewind" the Read Pointer for frame re-transmission due to a collision on the network.

TX FIFO Packet Counter (RO)

This 11-bit up/down counter keeps track of the number of frames that currently reside in the TxFIFO. The counter increments when a frame is loaded into the FIFO, and decrements when a frame has been transferred to the TX_MAC. This counter is used to enable frame transfer from the TxFIFO to the TX_MAC.

TX State Machine Register (RO)

This register provides the current state for all the state machines in TX.

Table 3-14 TX State Machine Register

Bits	Field Name	Description
[9:0]	Chain State	Describes the state of the Chaining state machine.
[11:10]	Checksum State	Describes the state of the Checksum state machine.
[17:12]	TX FIFO Load State	Describes the state of the TX FIFO Load State Machine. This field will have a value of 0x01 when the TX DMA is successfully disabled.
[21:18]	TX FIFO Unload State	Describes the state of the TX FIFO Unload State Machine.
[31:22]	RESERVED	

TX Data Pointer Low and High (RO)

A 64-bit pointer to the transmit data buffer in the host memory is stored in these two registers with the most significant bits in the High register. It is loaded by the DMA state machine, and it increments as the DMA reads in transmit data. Note that only the 41 least significant bits are incremented, while the upper 23 bits are taken as is from the TX descriptor.

TX FIFO Address (RW)

For diagnostic purposes a PIO path has been provided into the TX FIFO. This 11-bit register/counter holds the address of the 65 bit entry to be read/written. The value of this register increments on write accesses to either TX FIFO Data High register.

TX FIFO Tag, Data Low, Data HighT0, and Data HighT1 (RW)

These registers are used for diagnostics access to any TX FIFO location. Every access involves 65 bits. The 32 least significant bits are accessed through the TX FIFO Data Low, the 32 most significant bits through either TX FIFO Data High T0 or T1 registers, while the TX FIFO Tag register is dedicated to reading the tag bit. Writing via the TX FIFO Data HighT0 results in the tag bit being written low, and writing via the TX FIFO Data HighT1 sets the tag bit high.

A write sequence consists of:

- Writing the FIFO address into TX FIFO Address Register
- Writing the TX FIFO Data Low and one of the TX FIFO Data High Registers. The write operation to the Data High Register triggers the 65 bit FIFO write, and auto-increments the Address in preparation for the next cycle.

A read sequence consists of:

- Writing the FIFO address into TX FIFO Address Register

- Reading from any of the TX FIFO Tag, Data Low, and Data High Registers. The read operation does not auto-increment the Address.

Programming Restrictions: *Before proceeding with TXDMA FIFO PIO, follow the following sequence:*
1). Disable TxMAC.
2). Disable TxDMA.
3). Set the TX_FIFO_PIO_Sel bit of the TxDMA Configuration register.
To reverse from TXDMA FIFO PIO mode, follow the following sequence:
1). Reset the TX_FIFO_PIO_Sel bit.
2). Enable TxDMA.
3). Enable TxMAC.

TX FIFO Size (RO)

This 11-bit read only register indicates the size, in 64 byte multiples, of the TX FIFO. The value of this register is 0x090, indicating a 9kbyte TX FIFO.

TX Debug Register (RO)

32-bit register reflecting internal status.

3.1.4.3 RX DMA Programmable Resources

RX Configuration Register (RW)

This register stores parameters that control the receive DMA channel operation.

Table 3-15 RX Configuration Register

Bits	Field Name	Description
[0]	Rx_DMA_Enable	When set to '1', the RX DMA channel is enabled. When cleared to '0', the RX DMA channel will orderly stop operation as soon as the current frame transfer has been completed.
[4:1]	Rx_Desc_Ring_Size	This field determines the number of descriptor entries in the Rx ring. Default: 0x8; 8k descriptor entries.
[5]	Batch_Disable	When set to '1', disable receive descriptor "batching". Defaults low to batching enabled.
[9:6]	Reserved	

Table 3-15 RX Configuration Register

Bits	Field Name	Description
[12:10]	First_Byte_offset	This field determines the byte offset of the first data byte of the packet within 8 byte boundaries.
[19:13]	Checksum_Start_Offset	Checksum_Start_Offset. Indicates the number of bytes from the first byte of the packet that should be skipped before the TCP checksum calculation begins. Only even offsets are supported (bit 13 must be zero).
[26:24]	RX_DMA_threshold	When the RX FIFO does not contain a full RX frame, RX DMA will only request DMA transfers if the number of bytes in the RX FIFO exceeds MAX [RX_DMA_threshold, Cache Line]. Defaults to 128. Encoded as: 000 - 64 bytes 001 - 128 bytes 010 - 256 bytes 011 - 512 bytes 100 - 1024 bytes 101 - 2048 bytes 11X- Reserved

Table 3-16 RX Descriptor Ring Size values

Rx_Desc_Ring_Size	Ring Size
0	32
1	64
2	128
3	256
4	512
5	1k
6	2k
7	4k
8	8k
9-15	Reserved

Default: 0x1000010

RX Descriptor Base Low and High (RW)

The 53 most significant bits of this register are used as the base address for the RX descriptor ring. The 11 least significant bits are not stored, and assumed to be zero.

Programming Restrictions: *The Receive Descriptor Pointer must be initialized to a 2KByte-aligned value after power-on or software reset.*

RX FIFO Write Pointer (RW)

This 11-bit loadable counter points to the next location in the RxFIFO that will be loaded with data from the RX_MAC. The counter increments on 64-bit loads into the FIFO. The counter is loaded with the contents of Shadow Write Pointer, when an "early receive abort" needs to be performed. This counter generates the "write" address for the RxFIFO memory core.

RX FIFO Shadow Write Pointer (RW)

This 11-bit register points to the first word of the packet that is currently being loaded into the FIFO. The register is loaded with the contents of the Write pointer at the beginning of the packet transfer from the RX_MAC to the FIFO. This register is used to perform "early receive abort".

RX FIFO Read Pointer (RW)

This 11-bit loadable counter points to the next location in the RxFIFO that will be read from to retrieve packet data to be transferred to host memory. This counter generates the "read" address for the RxFIFO memory core, on 64-bit boundaries.

RX FIFO Packet Counter (RO)

This 12-bit up/down counter keeps track of the number of frames that currently reside in the RxFIFO. The counter increments after a frame is completely loaded into the FIFO, and decrements when a frame has been transferred to the host memory. This counter is used to enable a frame transfer to the host memory. The size of the counter is dimensioned for a worst case of 12 byte packets (error checking disabled and CRC stripped at MAC).

Default: 0x0000.

RX State Machine Register (RO)

This register provides the current state for the RX DMA state machines.

Default: 0x00000.

PAUSE Thresholds (RW)

Two PAUSE thresholds are used to define when PAUSE flow control frames are emitted by GEM. XOFF PAUSE frames use the pause_time value pre-programmed in the Send PAUSE MAC Register, while XON PAUSE frames use a pause_time of zero. The hysteresis mechanism for PAUSE frames is described in Section 2.2.4.2, "Sourcing of MAC Control Frames".

The granularity of these thresholds is in 64 byte increments. PAUSE thresholds

Table 3-17 PAUSE Thresholds

Bits	Field Name	Description
[8:0]	OFF threshold	XOFF PAUSE emitted when RX FIFO occupancy rises above this value (times 64 bytes).
[11:9]	Reserved	
[20:12]	ON threshold	XON PAUSE emitted when after emitting XOFF PAUSE, the RX FIFO occupancy falls below this value (times 64 bytes). Must be smaller than the OFF threshold value. If equal to the RX FIFO Size, XON frames are never emitted.

are defined in terms of FIFO occupancy, and may be translated into FIFO vacancy by the software driver using the RX FIFO Size register information.

Default: 0x000F8.

RX Data Pointer (RO)

A 64-bit pointer to the receive data buffer in the host memory is stored in these two registers with the most significant bits in the High register. It is loaded by the DMA state machine, and it increments as the DMA writes receive data. Note that only the 41 least significant bits are incremented, while the upper 23 bits are taken as is from the RX descriptor.

RX Kick Register (RW)

This 13-bit register written by the host CPU. The last valid receive descriptor is the one right before the value of the register. The *RX Kick Register* and *RX Completion Register* values are used by the receive DMA engine to control RX descriptor fetching and interrupt generation. If the value indicates that four or more valid RX descriptors are available, GEM will read four at time as needed, and internally cache them. Initialized to zero on reset.

Programming Restrictions: *Writing the value N in this register means that all descriptors up to but excluding N are available. Receive Descriptors must be posted in increments of four (N must be a multiple of four), and for best performance the first descriptor should be cache line aligned.*

RX Completion Register (RO)

This 13-bit register indicates which descriptors were already used by GEM for receive frames. All descriptors up to but excluding the register value are ready to be processed by the host. The DMA state machine updates the value of this register after the corresponding frame is entirely transferred from the RX FIFO into the host queue, but before setting the RX_DONE bit in the Interrupt Status Register. Initialized to zero on reset.

RX Blanking Register (RW)

This register defines the values used for receive interrupt blanking.

Table 3-18 RX Blanking Register

Bits	Field Name	Description
[8:0]	RX_INTR_PACKETS	RX_DONE interrupt will be asserted if that many packet are received since the last RX_DONE interrupt. A value of zero indicates no packet blanking.
[11:9]	Reserved	
[19:12]	RX_INTR_TIME	RX_DONE interrupt will be asserted if that many clocks were counted since the last RX_DONE interrupt. Every count is 2048 PCI clocks. A value of zero indicates no time blanking.

Default: 0x00000.

RX FIFO Address (RW)

For diagnostic purposes a PIO path has been provided into the RX FIFO. This 12-bit register / counter holds the address of the 65 bit entry to be read / written. The value of this register increments on write accesses to either RX FIFO Data High register.

RX FIFO Tag, Data Low, Data HighT0, and Data HighT1 (RW)

These registers are used for diagnostics access to any RX FIFO location. Every access involves 65 bits. The 32 least significant bits are accessed through the RX FIFO Data Low, the 32 most significant bits through either RX FIFO Data High T0 or T1 registers, while the RX FIFO Tag register is dedicated to reading the tag bit. Writing via the RX FIFO Data HighT0 results in the tag bit being written low, and writing via the RX FIFO Data HighT1 sets the tag bit high.

A write sequence consists of:

- Writing the FIFO address into RX FIFO Address Register
- Writing the RX FIFO Data Low and one of the RX FIFO Data High Registers. The write operation to the Data High Register triggers the 65 bit FIFO write, and auto-increments the Address in preparation for the next cycle.

A read sequence consists of:

- Writing the FIFO address into RX FIFO Address Register
- Reading from any of the RX FIFO Tag, Data Low, and Data High Registers. The read operation does not auto-increment the Address.

Programming Restrictions: *The RX_DMA_Enable bit must be set to zero for RX FIFO PIOs accesses.The Data High should be the last access of the sequence.*

RX FIFO Size (RO)

This 11-bit read only register indicates the size, in 64 byte multiples, of the RX FIFO. Software should use it to properly configure the PAUSE thresholds. The value read is 0x140, indicating a 20kbyte RX FIFO.

3.1.5 MAC Programmable Resources

3.1.5.1 Command Registers

These registers are used by the software to instruct the hardware that a certain hardware function is to be executed upon the detection of the command bit. The command bits are set to '1' using a PIO write, and are "self-cleared" after the command execution has completed. A command bit can be polled at any time using a PIO read to verify the command execution.

TX_MAC Software Reset Command (RW)

This 1-bit command performs a software reset to the logic in the TX_MAC. The bit is set to '1' when a Programmed IO write is performed to the defined address. This bit becomes "self-cleared" after the command has been executed.

Default: 0x0.

RX_MAC Software Reset Command (RW)

This 1-bit command performs a software reset to the logic in the RX_MAC. The bit is set to '1' when a Programmed IO write is performed to the defined address. This bit becomes "self-cleared" after the command has been executed.

Default: 0x0.

Send Pause Command (RW)

This command register executes a Pause Flow Control frame transmission.

[15:0]:	Pause_Time_Sent:	This control field indicates to the MAC the value of the pause_time operand that should be sent on the network using either the Send_Pause command bit or the flow control handshake on the RxDMA<->MAC interface. The pause_time is interpreted in units of Slot Times.
[16]:	Send_Pause:	This command bit is used by the software to instruct the MAC that a Pause Flow Control frame should be sent on the network.

Default: 0x0XXXX.

3.1.5.2 Status and Mask Registers

These registers are used by the hardware to communicate to the software the occurrence of events that were detected by the hardware. A status register may contain interrupt bits and information fields.
If an interrupt bit is set to '1', it indicates the occurrence of the corresponding event. Interrupt bits are auto-cleared when the status register is read by the software, and have corresponding mask bits in a mask register. If a mask bit is cleared to '0', the corresponding status event will generate an interrupt. Mask Registers default to all ones upon reset.

Information fields are provided to complement interrupt status bits. They are unaffected by PIO reads and cannot be masked.
Status bits default to all 0's, and the corresponding mask bits default to all 1's.

TxMAC Status Register (R-AC)

[0]: Frame_Transmitted: This interrupt status bit indicates the successful transmission of a frame on the network.

[1]: Tx_Underrun: This interrupt status bit indicates that the MAC has terminated a frame transmission due to "data starvation" in the transmit data path.

[2]: Max_Pkt_Err: This interrupt status bit indicates that a frame that exceeds the maximum allowed length was passed to the TxMAC by the DMA engine.

[3]: Normal_Coll_Cnt_Exp: This interrupt status bit indicates the roll over of the Normal Collision Counter.

[4]: Excess_Coll_Cnt_Exp: This interrupt status bit indicates the roll over of the Excessive Collision Counter.

[5]: Late_Coll_Cnt_Exp: This interrupt status bit indicates the roll over of the Late Collision Counter.

[6]: First_Coll_Cnt_Exp: This interrupt status bit indicates the roll over of the First Collision Counter.

[7]: Defer_Timer_Exp: This interrupt status bit indicates the roll over of the Defer Timer.

[8]: Peak_Attmpt_Cnt_Exp: This interrupt status bit indicates the roll over of the Peak Attempts Counter.

RxMAC Status Register (R-AC)

[0]: Frame_Received: This interrupt status bit indicates the successful reception of a frame from the network.

[1]: Rx_Overflow: This interrupt status bit indicates that the MAC has dropped a receive frame due to the lack of resources in the receive data path (RX FIFO overflow).

[2]: Frame_Cnt_Exp: This interrupt status bit indicates the roll over of the Receive Frame Counter.

[3]: Align_Err_Cnt_Exp: This interrupt status bit indicates the roll over of the Alignment Error Counter.

[4]: CRC_Err_Cnt_Exp: This interrupt status bit indicates the roll over of the CRC Error Counter.

[5]: Length_Err_Cnt_Exp: This interrupt status bit indicates the roll over of the Length Error Counter.

[6]: Viol_Err_Cnt_Exp: This interrupt status bit indicates the roll over of the Code Violation Error Counter.

MAC Control Status Register (R-AC)

[0]: Pause_Received: This interrupt status bit indicates the successful reception of a Pause Flow Control frame from the network.

[1]: Paused_State: This interrupt status bit indicates that the MAC has made a state transition from "not-paused" to "paused".

[2]: Not-Paused_State: This interrupt status bit indicates that the MAC has made a state transition from "paused" to "not-paused".

[15:3]: Reserved.

[31:16]: Pause_Time_Rcvd: This information field indicates the value of the "pause_time" operand that was received in the last Pause Flow Control frame.

After reset, the upper 16 bits default to "x" and the lower 16 bits default to zero.

TxMAC Mask Register (RW)

The layout of this register corresponds bit-by-bit to the layout of the TxMAC Status Register, bits [8:0].

RxMAC Mask Register (RW)

The layout of this register corresponds bit-by-bit to the layout of the RxMAC Status Register, bits [6:0].

MAC Control Mask Register (RW)

The layout of this register corresponds bit-by-bit to the layout of the MAC Control Status Register, bits [2:0].

3.1.5.3 Configuration Registers

XIF Configuration Register (RW)

This register determines the parameters that control the operation of the transceiver interface.

Table 3-19 XIF Configuration Register

Bits	Field Name	Description
[0]	Tx_MII_OE	When set to '1', this bit enables the output drivers on the MII transmit bus
[1]	MII_Int_Loopback	When this bit is set to '1', the MII/GMII transmit data path is internally (on-chip) looped back to the MII/GMII receive data path. The entire MAC core is driven off the system clock. Therefore the clock selection using the phy_mode register is ignored. This bit must be low for normal operation.
[2]	Disable_Echo	This bit should be enabled only in MII or GMII Half Duplex modes when an external PHY transceiver is used. When set to '1', this bit disables the receive data path during packet transmission by the TxMAC. Effectively, the RX_DV signal from the MII is forced to '0', when the TX_EN is active. This bit should be cleared to '0' when the MAC operates in the Full Duplex mode, or is in one of the Loopback modes (Internal or External), or is in Half Duplex Serdes or SLink modes.
[3]	GMIIMODE	When set it configures the MAC interface to operate with GMII clocks and datapath. When clear it operates with MII clocks and datapath.
[4]	MII_Buffer_OE	This bit controls the MII_BUF_EN pin. This bit when set is intended to enable the external tri-state buffer that may reside on the MII receive data bus. Multiple transceivers may be connected to the MAC using this mechanism. When this bit is '0' the pin is low and the buffer is disabled.
[5]	LINKLED	When set it forces the LINKLED# active (low).
[6]	FDPLXLED	When set it forces the FDPLXLED# active (low).

Default: 0x00.

Programming Restrictions: *To ensure proper operation of the hardware, when a loopback configuration is entered or exited, a Global Initialization Sequence should be performed.*

TX_MAC Configuration Register (RW)

This register controls the operation of the TX_MAC.

Table 3-20 TX_MAC Configuration Register

Bits	Field Name	Description
[0]	TxMAC_Enable	When set to '1', the TxMAC starts requesting packet data from the OPP, and the Ethernet transmit protocol execution begins. When cleared to '0', it will force the TxMAC state machines to either remain in the idle state, or to transition to the idle state and stay there at the completion of an ongoing packet transmission.
[1]	Ignore_Carrier_Sense	When set to '1', this bit causes the TxMAC to disable the CSMA/CD deferral process. This bit should be set to '1' while in Full Duplex mode, and cleared to '0' while in Half Duplex mode of operation.
[2]	Ignore_Collisions	When set to '1', this bit causes the TxMAC to disable the CSMA/CD backoff algorithm. This bit should be set to '1' while in Full Duplex mode, and cleared to '0' while in Half Duplex mode of operation.
[3]	Enable_IPG0	When set to '1', this bit enables the extension of the Rx-to-Tx IPG. After receiving a frame, the TxMAC will reset its deferral process in response to Carrier Sense assertion during the period of time that is the sum of the values programmed in the IPG0 and IPG1 registers, and will commit to transmission during the period of time that is programmed in the IPG2 register. When cleared to '0', or when transmitting frames back-to-back (Tx-to-Tx IPG), the TxMAC ignores IPG0. It will reset its deferral process in response to Carrier Sense assertion during the period of time that is programmed in the IPG1 register, and will commit to transmission during the period of time that is programmed in the IPG2 register.
[4]	Never_Give_Up	When set to '1', this bit modifies the CSMA/CD protocol, such that the TxMAC will not "easily give up" on a frame transmission. If the backoff algorithm reaches the attempts_limit, it will clear the attempts_counter and will continue trying to send the frame for an extended period of time (as defined by bit [5]). When this bit is cleared to '0', the TxMAC will execute the standard CSMA/CD protocol.

Table 3-20 TX_MAC Configuration Register

Bits	Field Name	Description
[5]	Never_Give_Up_Limit	When this bit is set to '1', the TxMAC continues trying to send out the frame until it is successfully transmitted on the medium. In effect, no limit will exist on the number of transmission attempts. When this bit is cleared to '0', the TxMAC will continue trying to transmit the frame until it is either successfully transmitted on the medium, or the backoff algorithm reached the attempts_limit for sixteen times.
[6]	No_Backoff	When set to '1', this bit modifies the CSMA/CD protocol, such that the backoff algorithm in the Protocol Engine is disabled. The TxMAC will not back off after a transmission attempt that resulted in a collision on the medium. Effectively, the random number, generated by the backoff algorithm, is always fixed to '0'. When this bit is cleared to '0', the TxMAC will execute the standard CSMA/CD protocol.
[7]	Slow_Down	When set to '1', this bit modifies the CSMA/CD protocol, such that the TxMAC resets its deferral process in response to the assertion of the Carrier Sense during the entire duration of the IPG. In effect, the TxMAC commits to the transmission of a frame only after the frame transmission has actually begun. When this bit is cleared to '0', the TxMAC will execute the standard CSMA/CD protocol.
[8]	No_FCS	When set to '1', this bit will force the TxMAC to not generate the CRC for all transmitted packets. If cleared to '0', the TxMAC will act on the CRC generation as indicated by the NO_CRC bit in the transmit Control Word from the TxDMA.
[9]	TX_Carrier_Extension	When set to '1', this bit enables the transmit part of the Carrier Extension feature. This feature allows longer collision domains by extending if necessary the carrier and the collision window from the end of the FCS until the end of the Slot Time. Carrier Extension is required for half-duplex operation at 1 Gbps, this bit should be clear otherwise.

Programming Restrictions: *To ensure proper operation of the TX_MAC, the TX_MAC_Enable bit must always be cleared to '0' and a delay imposed before a PIO write to any of the other bits in the TX_MAC Configuration register or any of the MAC parameters registers is performed.*

The amount of delay required will depend on the time required to transmit a maximum size frame, and is thus dependent on the value programmed into the MaxFrameSize register and the data rate on the medium. For a standard 1518-byte frame on a 100Mbps network the delay would be 125usec. To avoid the requirement for a variable time delay, the TX_MAC_Enable bit may be polled, and when this bit reads back as a '0', all the registers mentioned above may be written, including all the other bits in the Configuration register.

Note – Enabling Carrier Extension involves setting the TX_Carrier_Extension, RX_Carrier_Extension bits and configuring the Slot Time Register to 0x200 (Slot Time of 512 bytes). This mode must be enabled whenever GEM operates in Half-Duplex at 1Gbps, and disabled otherwise.

RX_MAC Configuration Register (RW)

This register controls the operation of the RX_MAC.

Table 3-21 RX_MAC Configuration Register

Bits	Field Name	Description
[0]	Rx_MAC_Enable	When set to '1', the RX_MAC is enabled. When cleared to '0', it will force the RX_MAC state machines to either remain in the idle state, or to transition to the idle state and stay there
[1]	Strip_Pad	When set to '1', this bit will cause the RX_MAC to strip the "pad" bytes of the receive frames
[2]	Strip_FCS	When set to '1', this bit will cause the RxMAC to strip the last four bytes of a received frame.
[3]	Promiscuous	When set to '1', this bit will cause the RX_MAC to accept all valid frames from the network, regardless of the contents of the DA field of a frame.
[4]	Promiscuous_Group	When set to '1', this bit will cause the RxMAC to accept all valid multicast frames (frames that have the "group" bit in the DA field set to '1').
[5]	Hash_Filter_Enable	When set to '1', the RX_MAC will use the Hash Table to filter multicast addresses

Table 3-21 RX_MAC Configuration Register

Bits	Field Name	Description
[6]	Address_Filter_Enable	When set to '1', this bit will cause the RxMAC to use the Address Filtering Registers to filter both unicast and multicast addresses.
[7]	Disable_Discard_on_Err	When set to '1', this bit will cause the RxMAC to pass errored frames to the RX DMA by setting the BAD bit but not the Abort bit in the status. Frame validity checking for CRC, framing and length errors will still increment error counters. Frames will still be accepted or rejected based on address checking only.
[8]	RX_Carrier_Extension	When set to '1', this bit enables the receive part of the Carrier Extension feature. This feature enables the reception of packet bursts generated by Carrier Extension with Packet Bursting senders. This mode only applies for half-duplex at 1Gbps. This bit should be cleared otherwise.

Programming Restrictions:

1. To ensure proper operation of the RX_MAC, the RX_MAC_Enable bit must always be cleared to '0' and a delay of 3.2msec imposed before a PIO write to any of the other bits in the RX_MAC Configuration register or any of the MAC parameters' registers is performed. To avoid the requirement for a fixed time delay, the RX_MAC_Enable bit may be polled, and when this bit reads back as a '0', all the registers mentioned above may be written, including other bits in the Configuration register.

2. To ensure proper operation of the RX_MAC, the Hash_Filter_Enable bit in the RX_MAC Configuration register must always be cleared to '0' and a delay of 3.2msec imposed before a PIO write to any of the Hash Table registers is performed. To avoid the requirement for a fixed time delay, the Hash_Filter_Enable bit may be polled, and when this bit reads back as a '0', all the registers mentioned above may be written.

3. To ensure proper operation of the RX_MAC, the Address_Filter_Enable bit in the RX_MAC Configuration register must always be cleared to '0' and a delay of 3.2msec imposed before a PIO write to any of the Address Filter registers is performed. To avoid the requirement for a fixed time delay, the Address_Filter_Enable bit may be polled, and when this bit reads back as a '0', all the registers mentioned above may be written.

MAC Control Configuration Register (RW)

[0]: Send_Pause_Enable: When set to '1', this bit enables the MAC to start responding to requests for sending Pause Flow Control frames.
If cleared to '0', the MAC will ignore both the software Send_Pause commands and the flow control handshake requests on the RxDMA<->MAC interface.

[1]: Receive_Pause_Enable: When set to '1', this bit enables the MAC to start responding to received Pause Flow Control packets.
If cleared to '0', the MAC will ignore all received Pause Flow Control packets.

[2]: Pass_MAC_Control: When set to '1', this bit enables the RxMAC to pass valid MAC Control packets to the RxDMA.
If cleared to '0', the RxMAC will "consume" the MAC Control frame, and will not pass it to RxDMA.

Default: 0x0.

3.1.5.4 Protocol Parameters Registers

Inter Packet Gap 0 Register (RW)

The value in this 8-bit register is used as an extension of the value in the Inter Packet Gap 1 Register for timing the Receive-to-Transmit IPG. The TxMAC will ignore this value and presume it to be equal to zero for the purpose of timing the Transmit-to-Transmit IPG, and/or when the Enable_IPG0 bit in the TxMAC Configuration Register is cleared to '0'.
The time interval specified in this register is in units of media byte time.

Recommended value: 0x00.

Inter Packet Gap 1 Register (RW)

The value in this 8-bit register defines the first 2/3 portion of the Inter Packet Gap (IPG), which is timed by the TxMAC before each frame's transmission is initiated.

For back-to-back transmissions (Transmit-to-Transmit IPG), this value is added to the value in the Inter Packet Gap 2 Register, and during the entire period of time the CarrierSense indication (if present) is ignored by the TxMAC.

For a frame reception followed by a frame transmission (Receive-to-Transmit IPG), the TxMAC will monitor the CarrierSense indication, and (if present) will reset its deferral process during the time interval defined by the sum of values in this register and in the Inter Packet Gap 0 Register, but will ignore it during the time interval specified in the Inter Packet Gap 2 Register.

The time interval specified in this register is in units of media byte time.

Recommended value: 0x08.

Inter Packet Gap 2 Register (RW)

The value in this 8-bit register determines the second 1/3 portion of the Inter Packet Gap parameter.
The time interval specified in this register is in units of media byte time.

Recommended value: 0x04.

Slot Time Register (RW)

This 10-bit register specifies the slot time parameter in units of media byte time. This parameter determines the physical span of the network.

Recommended value: 0x40.

Minimum Frame Size Register (RW)

This 10-bit register determines the minimum number of bytes that the TxMAC will transmit for any frame on the medium, and the RxMAC will receive from the medium before it recognizes the frame to be valid.

Recommended value: 0x40.

Maximum Frame and Burst Size Register (RW)

This 30-bit register specifies the maximum number of bytes that the TxMAC will transmit for any frame on the medium, and the RxMAC will receive from the medium before it recognizes the frame to be not valid. It also specifies the maximum run length of a burst of packets sent in the Half Duplex mode (only relevant for Gigabit Ethernet operation).
[30:16] --- Maximum Burst Size.
[15] --- Reserved.
[14:0] --- Maximum Frame Size.

Recommended value: 0x200005EE.

PA Size Register (RW)

This 10-bit register specifies the number of Preamble bytes that the TxMAC will transmit at the beginning of each frame. The value programmed in this register should be two or greater.

Recommended value: 0x07.

Jam Size Register (RW)

This 4-bit register specifies the duration of the jam in units of media byte time.

Recommended value: 0x04.

Attempts Limit Register (RW)

This 8-bit register specifies the number of attempts that the TxMAC will make to transmit a frame, before it resets its Attempts_Counter. After reaching the Attempts_Limit, the TxMAC may or may not drop the frame, as determined by the NGU and NGUL bits in the TxMAC Configuration Register.

Recommended value: 0x10.

MAC Control Type Register (RW)

This 16-bit register specifies the "Type" field of a MAC Control frame.
The TxMAC uses this field to encapsulate a MAC Control frame for transmission, and the RxMAC uses it for decoding valid MAC Control frames received from the network.

Recommended value: 0x8808.

3.1.5.5 Address Detection and Filtering Registers

MAC Address 0 Register (RW)

This 16-bit register contains the most significant bits of a station's normal priority MAC Address. These bits will be compared against bits [47:32] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be a unicast address.

MAC Address 1 Register (RW)

This register contains the 16 middle bits of a station's normal priority MAC Address. These bits will be compared against bits [31:16] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be a unicast address.

MAC Address 2 Register (RW)

This register contains the 16 least significant bits of a station's normal priority MAC Address. These bits will be compared against bits [15:0] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be a unicast address.

MAC Address 3 Register (RW)

This register contains the 16 most significant bits of the station's alternate MAC Address. These bits will be compared against bits [47:32] of the DA field for every frame that arrives from the network. The MAC address programmed in this register may be a unicast or a multicast address.

MAC Address 4 Register (RW)

This register contains the 16 middle bits of the station's alternate MAC Address. These bits will be compared against bits [31:16] of the DA field for every frame that arrives from the network. The MAC address programmed in this register may be a unicast or a multicast address.

MAC Address 5 Register (RW)

This register contains the 16 least significant bits of the station's alternate MAC Address. These bits will be compared against bits [15:0] of the DA field for every frame that arrives from the network. The MAC address programmed in this register may be a unicast or a multicast address.

MAC Address 6 Register (RW)

This register contains the 16 most significant bits of the MAC Control Address. These bits will be compared against bits [47:32] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be the reserved multicast address for MAC Control frames.

MAC Address 7 Register (RW)

This register contains the 16 middle bits of the MAC Control Address. These bits will be compared against bits [31:16] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be the reserved multicast address for MAC Control frames.

MAC Address 8 Register (RW)

This register contains the 16 least significant bits of the MAC Control Address. These bits will be compared against bits [15:0] of the DA field for every frame that arrives from the network. The MAC address programmed in this register must be the reserved multicast address for MAC Control frames.

Note – A MAC Address of the form a:b:c:d:e:f where bytes **abc** comprise the OUI part of the address and bytes **def** are the vendor unique serial number would be stored as shown Table 3-22. The Group Address bit is the least significant bit of byte **a**.

Table 3-22 MAC Address Notation a:b:c:d:e:f

	ab	**cd**	**ef**
Station Unique Address	MAC Address 2 Register	MAC Address 1 Register	MAC Address 0 Register
Alternate Address	MAC Address 5 Register	MAC Address 4 Register	MAC Address 3 Register
MAC Control Address	MAC Address 8 Register	MAC Address 7 Register	MAC Address 6 Register

Address Filter 0 Register (RW)

This 16-bit register contains bits [47:32] of the Address Filter.

Address Filter 1 Register (RW)

This 16-bit register contains bits [31:16] of the Address Filter.

Address Filter 2 Register (RW)

This 16-bit register contains bits [15:0] of the Address Filter.

Address Filter 2&1 Mask Register (RW)

This 8-bit register contains a nibble mask for Address Filter Registers 2 and 1.

Address Filter 0 Mask Register (RW)

This 16-bit register contains a bit mask for the Address Filter Register 0.

Hash Table 0 Register (RW)

This 16-bit register contains bits [255:240] of the Hash Table.

Hash Table 1 Register (RW)

This 16-bit register contains bits [239:224] of the Hash Table.

Hash Table 2 Register (RW)

This 16-bit register contains bits [223:208] of the Hash Table.

Hash Table 3 Register (RW)

This 16-bit register contains bits [207:192] of the Hash Table.

Hash Table 4 Register (RW)

This 16-bit register contains bits [191:176] of the Hash Table.

Hash Table 5 Register (RW)

This 16-bit register contains bits [175:160] of the Hash Table.

Hash Table 6 Register (RW)

This 16-bit register contains bits [159:144] of the Hash Table.

Hash Table 7 Register (RW)

This 16-bit register contains bits [143:128] of the Hash Table.

Hash Table 8 Register (RW)

This 16-bit register contains bits [127:112] of the Hash Table.

Hash Table 9 Register (RW)

This 16-bit register contains bits [111:96] of the Hash Table.

Hash Table 10 Register (RW)

This 16-bit register contains bits [95:80] of the Hash Table.

Hash Table 11 Register (RW)

This 16-bit register contains bits [79:64] of the Hash Table.

Hash Table 12 Register (RW)

This 16-bit register contains bits [63:48] of the Hash Table.

Hash Table 13 Register (RW)

This 16-bit register contains bits [47:32] of the Hash Table.

Hash Table 14 Register (RW)

This 16-bit register contains bits [31:16] of the Hash Table.

Hash Table 15 Register (RW)

This 16-bit register contains bits [15:0] of the Hash Table.

3.1.5.6 Statistics Registers

Normal Collision Counter (RW)

This 16-bit loadable counter increments for every frame transmission attempt that resulted in a collision.

Recommended initialization to: 0x0000.

First Attempt Successful Collision Counter (RW)

This 16-bit loadable counter will increment for every frame transmission that experienced a collision on the first attempt, but was successfully transmitted on the second attempt.

Recommended initialization to: 0x0000.

Excessive Collision Counter (RW)

This 16-bit loadable counter increments for every frame transmission that has exceeded the Attempts Limit.

Recommended initialization to: 0x0000.

Late Collision Counter (RW)

This 16-bit loadable counter increments for every frame transmission that has experienced a late collision. It indicates the number of frames that the TxMAC has dropped due to collisions that occurred after it has transmitted at least the Minimum Frame Size number of bytes. Usually this is an indication that there is at least one station on the network that violates the maximum allowed span of the network.

Recommended initialization to: 0x0000.

Defer Timer (RW)

This 16-bit loadable timer increments when the TxMAC is deferring to traffic on the network while it is attempting to transmit a frame. The time base for the timer is the media byte clock divided by 256. Thus, on a 10Mbps network the timer ticks are 200 *m*sec, and on a 100Mbps network the timer ticks are 20*m*sec. Software may clear it by writing zeros to the timer.

Recommended initialization to: 0x0000.

Peak Attempts Register (R-AC)

This 8-bit register indicates the highest number of consecutive collisions per successfully transmitted frame, that have occurred since this register was last read. The maximum value that this register can attain is 255. A maskable interrupt is generated to the software if the number of consecutive collisions per successfully transmitted frame exceeds 255. This register will automatically be cleared to '0' after it is read.
Recommended initialization to: 0x00.

Receive Frame Counter (RW)

This 16-bit loadable counter increments after a valid frame has been received from the network.

Recommended initialization to: 0x0000.

Length Error Counter (RW)

This 16-bit loadable counter increments after a frame, whose length is greater than the value that was programmed in the Maximum Frame Size Register, has been received from the network.

Recommended initialization to: 0x0000.

Alignment Error Counter (RW)

This 16-bit loadable counter increments when an alignment error was detected in a receive frame. An alignment error is reported when a receive frame fails the CRC checking algorithm, AND the frame contains a non-integer number of bytes (i.e. the frame size in bits modulo 8 is not equal to zero).

Recommended initialization to: 0x0000.

FCS Error Counter (RW)

This 16-bit loadable counter increments when a receive frame fails the CRC checking algorithm, AND the frame contains an integer number of bytes (i.e. the frame size in bits modulo 8 is equal to zero).

Recommended initialization to: 0x0000.

Rx Code Violation Counter (RW)

This 16-bit loadable counter increments when a Rx_Err indication is generated by the XCVR over the MII, while a frame is being received. This indication is generated by the transceiver when it detects an invalid code in the received data stream. A receive code violation is not counted as an FCS or an Alignment error.

Recommended initialization to: 0x0000.

3.1.5.7 Miscellaneous Registers

Random Number Seed Register (RW)

This 10-bit register is used as a seed for the random number generator for the CSMA/CD backoff algorithm. The register has significance only after power-on reset, and should be programmed with a random value which has a high likelihood of being unique for each MAC attached to a network segment (e.g. 10 LSB of the MAC address). During normal operation, the register contents are constantly updated by the hardware with new "random" values. Therefore, a PIO read from this register will return an unpredictable result.

Pause Timer (RO)

This 16-bit timer is used to time the pause interval as indicated by a received pause flow control frame. A non-zero value in this timer indicates that the MAC is currently in the paused state.

Default: 0x000.

TxMAC State Machine Register (RO)

This 8-bit register provides the current state for all the state machines in the Tx-MAC.

Default: 0x00.

RX_MAC State Machine Register (RO)

This 3-bit register provides the current state for all the state machines in the Rx-MAC.

Default: 0x0.

3.1.5.8 MIF Programmable Resources

MIF Bit-Bang Clock (RW)

This 1-bit register is used to generate the MDC clock waveform on the MII Management Interface when the MIF is programmed in the "Bit-Bang" Mode. Writing a '1' after a '0' into this register will create a rising edge on the MDC, while writing a '0' after a '1' will create a falling edge. For every bit that is transferred on the management interface, both edges have to be generated.

MIF Bit-Bang Data (RW)

This 1-bit register is used to generate the outgoing data (MDO) on the MII Management Interface when the MIF is programmed in the "Bit-Bang" Mode. The data will be steered to the appropriate MDIO based on the state of the PHY_Select bit in the MIF Configuration Register.

MIF Bit-Bang Output Enable (RW)

This 1-bit register is used to enable ('1') and disable ('0') the I-directional driver on the MII Management Interface when the MIF is programmed in the "Bit-Bang" Mode. The MDIO should be enabled when data bits are transferred from the MIF to the transceiver, and it should be disabled when the interface is idle or when data bits are transferred from the transceiver to the MIF (data portion of a read instruction). Only one MDIO will be enabled at a given time, depending on the state of the PHY_Select bit in the MIF Configuration Register.

MIF Configuration Register (RW)

This 15-bit register controls the operation of the MIF.

Table 3-23 MIF Configuration Register

Bits	Field Name	Description
[0]	PHY_Select	The MIF implements two independent management interfaces for two separate transceivers. Only one transceiver can be used at a given time. This bit determines which transceiver is currently in use. When cleared to '0', MDIO_0 is selected. Went set to '1', MDIO_1 is selected.
[1]	Poll_Enable	When set to '1', this bit enables the polling mechanism as described in 2.4.6. If this bit is set to '1', the BB_Mode should be cleared to '0'
[2]	BB_Mode	This bit determines the mode of operation of the MIF. When set to '1', the "bit-bang mode" is selected. When cleared to '0', the "frame mode" will be used.
[7:3]	Poll_Reg_Addr	This field determines the register address in the transceiver that will be polled by the polling mechanism in the MIF. It is meaningful only if the Poll_Enable bit is set to '1'

Table 3-23 MIF Configuration Register

Bits	Field Name	Description
[8]	MDI_0	This read-only bit is dual-purpose. **When the MDIO_0 interface is idle**, this bit will indicate whether a transceiver is connected to this line. If this bit reads as '1', the transceiver is connected. When the MIF is communicating with a transceiver that is hooked up to MDIO_0 in the bit-bang mode, this bit will indicate the incoming bit stream during a read operation
[9]	MDI_1	This read-only bit is dual-purpose. **When the MDIO_1 interface is idle**, this bit will indicate whether a transceiver is connected to this line. If this bit reads as '1', the transceiver is connected. When the MIF is communicating with a transceiver that is hooked up to MDIO_1 in the bit-bang mode, this bit will indicate the incoming bit stream during a read operation
[14:10]	Poll_Phy_Addr	This field determines the transceiver address to be polled

MIF Frame/Output Register (RW)

This 32-bit register serves as an "Instruction Register" when the MIF is programmed in the Frame Mode. In order to execute a read/write operation from/to a transceiver register, the software has to load this register with a valid instruction, as per IEEE 802.3u MII specification. After issuing an instruction, the software has to poll this register to check for instruction execution completion. During a read operation, this register will also contain the 16-bit data that was returned by the transceiver.

Table 3-24 MIF Frame/Output Register

Bits	Field Name	Description
[31:30]	ST	STart of frame. When issuing an instruction: This field should always be loaded with a '01'. When polling for completion: This field is always a "don't care".
[29:28]	OP	OPcode. When issuing an instruction: This field should be loaded with '01' for a "write" and with '10' for a "read". When polling for completion: This field is always a "don't care".
[27:23]	PHYAD	PHY ADdress. When issuing an instruction: This field should be loaded with the XCVR address. When polling for completion: This field is always a "don't care".

Table 3-24 MIF Frame / Output Register

Bits	Field Name	Description
[22:18]	REGAD	REGister ADdress. When issuing an instruction: This field should be loaded with the address of the register that is to be read / written. When polling for completion: This field is always a "don't care".
[17]	TA_MSB	Turn Around, Most Significant Bit. When issuing an instruction: This bit should always be loaded with a '1'. When polling for completion: This bit is always a "don't care".
[16]	TA_LSB	Turn Around, Least Significant Bit. When issuing an instruction: This bit should always be loaded with a '0'. When polling for completion: This bit serves as a "Valid Bit". When this bit is set to'1', the instruction execution has been completed.
[15:0]	DATA	Instruction Payload. When issuing an instruction: This field should be loaded with the 16-bit data to be written into a transceiver register for a "write", and is a "don't care" for a "read". When polling for completion: This field is a "don't care" for a "write", and contains the 16-bit data returned by the transceiver for a "read" (if the Valid Bit is set).

MIF Status Register (R-AC)

This 32-bit register is used in conjunction with the Poll Mode in the MIF. It contains two portions: Poll Data and Poll Status. The Poll Data field will always contain the latest "image update" of the XCVR register that is being polled, while the Poll Status field will indicate which bits in the Poll Data field have changed since the MIF Status Register was last read. The Poll Status field is "auto-cleared" after being read.

Table 3-25 MIF Status Register

Bits	Field Name	Description
[31:16]	Poll_Data	
[15:0]	Poll_Status	

MIF Mask Register (RW)

This 16-bit register is used to determine which bits in the Poll Status portion of the MIF Status Register will cause an interrupt. If a mask bit is cleared to '0', the corresponding bit of the Poll Status will generate the MIF Interrupt when set.

Default: 0xFFFF.

MIF State Machine Register (RO)

This 7-bit register provides the current state for all the state machines in the MIF.

3.1.6 PCS Programmable Resources

PCS MII Control Register (RW)

This register is equivalent to the Control Register of the standard MII management register set. Unlike MII management registers, this register is directly mapped in GEM's register space.

Table 3-26 PCS MII Control Register

Bits	Field Name	Description
[5:0]	Reserved	
[6]	1000 Mb/s Speed Select	Reads back one. Ignored on writes.
[7]	Collision Test	When set the COL signal at the PCS to MAC interface is activated regardless of activity on the receive inputs.
[8]	Duplex Mode	Forced low. PCS behavior is similar for half and full duplex.
[9]	Restart Auto-Negotiation	Self clearing bit. Set to restart Auto-Negotiation.
[10]	Isolate	Reads back as zero. Ignored on writes.
[11]	Power Down	Reads back as zero. Ignored on writes.
[12]	Auto-Negotiation Enable	When set the PCS goes through an automatic link configuration phase before it can be used. When clear the link can be used without any link configuration phase. Defaults high.
[13]	10/100 Speed Selection	Reads back as zero. Ignored on writes.
[14]	Wrapback	When set, loopback is enabled at the 10-bit interface (input side of Serialink). Clear for normal operation. It is recommended that when the Serialink is programmed for wrapback, that the LOCKREF bit be set in the Serialink Control register.
[15]	Reset	Resets the PCS when set. Self-clearing when reset process is complete.

Default: 0x1040.

Note – The Auto-Negotiation Enable bit should be programmed the same at the Link Partner as in the local device. Otherwise, Auto-Negotiation will not complete. When this bit is reprogrammed, Auto-Negotiation/Manual-Configuration is restarted automatically.

PCS MII Status Register (RO)

This register is equivalent to the Status Register of the standard MII management register set. Unlike MII management registers, this register is directly mapped in GEM's register space.

Table 3-27 PCS MII Status Register

Bits	Field Name	Description
[0]	Extended Capability	Reads zero.
[1]	Jabber Detect	Reads zero.
[2]	Link Status	1=Link is up; 0=Link is down This bit incorporates a latch on 0, such that the 0 is kept until read. The register may be read twice to determine if the link has gone up again.
[3]	Auto-Negotiation Ability	Always reads 1 (Able to perform Auto-negotiation)
[4]	Remote Fault	When set, this bit indicates that a Remote Fault has been detected from the received Link Code Word. Only valid after completion of Auto-Negotiation.
[5]	Auto-Negotiation Complete	1=Auto-Negotiation complete 0=Auto-Negotiation not completed
[7:6]	Reserved	
[8]	Extended Status	This bit can be used as an indication that this is a 1000 Base-X PHY. Reads back one. Writes to it are ignored.
[15:9]	Reserved.	

Default: 0x0108.

PCS MII Advertisement Register (RW)

This register is equivalent to the Advertisement Register of the standard MII management register set. Unlike MII management registers, this register is directly mapped in GEM's register space. The contents of this register are used during Auto-Negotiation.

Table 3-28 PCS MII Advertisement Register

Bits	Field Name	Description
[4:0]	Reserved	
[5]	Full Duplex	Advertises device's capability to perform full duplex 1000 Base-X.
[6]	Half Duplex	Advertises device's capability to perform half duplex 1000 Base-X.
[7]	PAUSE	Advertises device's capability to perform PAUSE symmetrical to its link partner.
[8]	ASM_DIR	Advertises device's capability to perform PAUSE asymmetric to its link partner.

Table 3-28 PCS MII Advertisement Register

Bits	Field Name	Description
[11:9]	Reserved	
[13:12]	Remote Fault	Provides simple fault information to the link partner. Bit 13 is writable by software to optionally indicate to its link partner that it is going off-line. Bit 12 will get set when signal_detect equals FAIL, and will remain set until successful negotiation at which time it will be set to zero.
[14]	Ack	Read Only.
[15]	Next Page	Read Only. Forced low.

Default: 0x00E0.

PCS MII Link Partner Ability Register (RW)

This register is equivalent to the Link Partner Ability Register of the standard MII management register set. Unlike MII management registers, this register is directly mapped in GEM's register space. The contents of this register are updated as a result of Auto-Negotiation. The register layout and bit definitions correspond to the PCS MII Advertisement Register.

PCS Configuration Register (RW)

Table 3-29 PCS Configuration Register

Bits	Field Name	Description
[0]	Enable	Enables the PCS functions. Must be zero while modifying the PCS MII Advertisement Register. To isolate the MAC from the media, one can use this bit by programming it to 0 and then setting the restart auto-negotiation bit in the MII Control register. The lonk will go down and not come up until this bit is set to 1.
[1]	Signal_detect override	Sets signal_detect internally to OK in case the optic is not able to generate a reliable signal.
[2]	Signal_detect active low	When set, changes the interpretation of the incoming optic signal so that when the signal is low, signal_detect is OK.
[4:3]	jitter_study	Allows detailed jitter measurements to be made to characterize optic parts. A single code group is transmitted repeatedly. Disparity rules are followed. Program to zero for normal operation. 0x0 = normal operation 0x1 = high frequency test pattern, D21.5 0x2 = low frequency test pattern, K28.7 0x3 = reserved
[5]	10 ms timer override	Shortens the 10-20 ms timer used in Auto-Negotiation and synchronization to a few cycles. The purpose is to shorten ASIC simulation time and vector generation. This should be programmed to zero for interoperability with other devices.

Default: 0x0.

PCS State Machine Register (RO)

Default: 0x00000

PCS Interrupt Status Register (R-AC)

This register indicates interrupt changes in specific PCS MII Status bits. Presently only one bit is implemented, reflecting transitions on the link status. The rationale for a single bit is that all other relevant bits cannot change without altering the link state. There is no mask register at this level. The PCS_INT bit may be masked at the Interrupt Status Register level.

Table 3-30 PCS Interrupt Status Register

Bits	Field Name	Description
[2]	Link Status Change	When set it indicates the Link Status has changed at least once since this register was read.

3.1.7 Serialink Programmable Resources

Datapath Mode Register (RW)

This register controls which network interface is used. No more than one bit should be set in this register.

Table 3-31 Datapath Mode Register

Bits	Field Name	Description
[0]	Serialink Mode	When set to '1', the internal Serialink is selected.
[1]	External SERDES Mode	When set to '1' the PCS is used via the 10-bit interface.
[2]	MII/GMII Mode	When set to '1' the PCS is not used, and the MII/GMII interface is selected. Selection between MII and GMII is controlled by the XIF register.
[3]	GMIIOUTEN	Only applicable to Serialink Mode. Setting this bit makes the 10-bit transmit data is visible at the GMII. When clear these outputs are disabled.

Default: none.

Serialink Control Register (RW)

This register controls inputs to the Serialink block.

Table 3-32 Serialink Control Register

Bits	Field Name	Description
[0]	LOOPBACK/ LOOPBACKN	When the PMA in use is the Serialink, loopback is enabled at the serial interface when this bit is set. The Serialink receive inputs are isolated from the Serialink core inputs during loopback. When the PMA in use is an external Serdes, loopback is enabled when this bit is set to zero.
[1]	ENSYNCDET	Enable Sync Character Detection. Setting this bit enables the receiver to detect the sync character in the received data. When clear sync character detection is automatically disabled once the receiver establishes synchronization. Should be low for normal operation.
[2]	LOCKREF	Lock to reference clock. When set, it causes the receiver to frequency-lock RBC[0:1] to the reference clock REFCLK. When clear, the receiver's clock generator locks to incoming serial data. This bit should be set for parallel wrapback mode (see Table 3-35).

Table 3-32 Serialink Control Register

Bits	Field Name	Description
[4:3]	EMP	These bits control the output driver emphasis. 00-no emphasis Values different than 00 may be used to reduce data dependent jitter by increasing the current added to certain bit patterns. The use of these bits depends on the transmission medium characteristics.
[5]	Reserved	Forced low.
[8:6]	SELFTEST	These bits are used for selecting different built in self test bit patterns. Must be all zeros for normal operation.
[9]	SW_PDOWN	Software power down for the Serialink block. Active high.
[11:10]	RX_ZERO	Programmable PLL input to Serialink
[13:12]	RX_POLE	Programmable PLL input to Serialink
[15:14]	TX_ZERO	Programmable PLL input to Serialink
[17:16]	TX_POLE	Programmable PLL input to Serialink

Default: 0x000

Shared Output Select Register (RW)

This register controls the multiplexing of test outputs into the PROM address (PA_3 through PA_0) pins. Should be set to 0x0 for normal operation.

Table 3-33 Datapath Mode Register

Bits	Field Name	Description
[1:0]	PROM_Addr	00 - Normal operation, PROM addresses [3:0] are selected. 01 - {PROM addresses [3:2], rx_tag, tx_tag} are selected. 10 - Serialink test outputs {passn,rxlockr,txlockr,recclk} are selected. 11 - Undefined.

Default: 0x0.

Serialink State Register (RO)

This register indicates the progress of the Serialink boot up. Used for diagnostic purposes, may be ignored by normal software drivers.

Default: 0x0.

3.1.8 PROM Image

The PROM image mapped in GEM's register space provides an additional mechanism for reading the PROM, and unlike the space programmed in the Expansion ROM Base Address Register, it may be used by the software driver at any time.

3.2 Programming Notes

3.2.1 Initialization Sequence

This section describes GEM's software initialization sequence for typical environments. Register initialization consists mostly of writing constants to the relevant registers, except for a few cases where the value to be written is determined during run-time as a function of:

1. host system characteristics / driver configuration files
2. values read from other GEM registers
3. values read from PHY devices and interface pins

The basic initialization flow is shown in Figure 3-1, and the detailed guidelines for each step are discussed below. This flow assumes a full blown GEM environment (NIC or motherboard) including at least one PHY device. Simpler diagnostic environments might omit some of the steps.

A) After hardware reset, GEM's PCI Configuration Space must be initialized. This is usually done by the platform OBP or BIOS function, with no driver involvement. This process must initialize the following GEM PCI Configuration Space Registers:

- Command Register - Bits that must be enabled: Memory Space, Bus Master. Recommended if possible: MWI Enable, Parity Error Response.
- Cache Line Size.
- Base Address Register.
- Interrupt Line - Not interpreted by GEM, but this information might be needed by the driver.

B) At this point GEM's register space is accessible to the driver. The driver can determine what type of PHY interface to use in terms of pin sharing. This can be done just once if the driver stores the result for future use after software reset. A viable algorithm for interface selection is:

- Read the value of MDIO0 and MDIO1(MIF Configuration Register). If either one is high, an MII PHY is present. Assume MII interface will be used.
- Otherwise, unless the driver has configuration information selecting the SERDES mode, assume the Serialink will be used.

In addition, the Random Number Seed Register should be written at this time with a 10 bit value uncorrelated across different nodes.

C) Using the values determined in B) or F) proceed:

- Datapath Mode Register - Set one of Serialink, Serdes, or MII / GMII.
- If MII, determine management interface to use (MDIO1 corresponds to external transceiver and has precedence over MDIO0), take PHY out of isolate mode, and initialize PHY. Enable MII PHY Auto-negotiation.
- Initialize and enable Serialink except in SERDES mode.
- Initialize and enable the PCS.

D) Set up the TX and RX DMA configuration by programming:

- TX Descriptor Ring Size and Base -
- TX FIFO Threshold -
- RX Descriptor Ring Size and Base -
- RX DMA Threshold -
- Pause Thresholds -
- RX Blanking -

Note – For software compatibility with future variants of GEM, the OFF threshold and ON threshold in the PAUSE Thresholds Register should be programmed considering the value read from the RX FIFO Size Register. For lossless flow control the OFF threshold should be between 4.5 to 6kbytes below the RX Fifo Size. This value depends on the link length. It is possible to make the OFF threshold adaptive starting with an arbitrary value and lowering the threshold if RX FIFO Overflows occur. A simple way of disabling PAUSE frame emission is to program the OFF threshold to the full RX FIFO Size.

E) Set up the MAC by programming a sane set of values. Note that most MAC registers power up undefined. Recommended values for these MAC registers are given in Table 3-34.

F) Wait for Link UP.

If Datapath is set for MII wait for MII transceiver link up. If there is no MII link but there is PCS link, go back to C) with a revised Datapath value.

If Datapath was not set for MII, wait until PCS link goes up.

The critical link negotiated values are only valid when the link is up (Auto-negotiation succeeded). Either stay here until link is up, or enable the appropriate Link status change interrupt and wake up when it is done.

Verify link is still up, enable interrupt to detect if it goes down. Save the values of:

- Full Duplex -
- Flow Control -
- Carrier Extension -
- Speed -

G) Re-program MAC to match the values found in F). This involves registers:

- TX_MAC Configuration Register - Bits: Ignore_Carrier_Sense, Ignore_Collisions.
- MAC Control Configuration Register - Bits: Send_Pause_Enable, Receive_Pause_Enable.
- Slot Time Register - 64 bytes normally. 512 bytes for half duplex @1Gbps. speed.
- XIF Configuration Register - GMIIMODE, set for 1Gbps, clear otherwise, clear MII_Int_Loopback

H) Enable:

- TX Configuration Register - Set the Tx_DMA_Enable bit. Other bits not modified.

- RX Configuration Register - Set the Rx_DMA_Enable bit. Other bits not modified.
- TX_MAC Configuration Register - Set the TxMAC_Enable bit. Other bits not modified.
- RX_MAC Configuration Register - Set the RxMAC_Enable bit. Other bits not modified.
- Interrupt Mask Register - Clear all bits corresponding to desired interrupts, possible bits to enable: TX_ALL, RX_DONE, Rx_Buffer_Not_Available, TX_MAC_INT, RX_MAC_INT, and either the MIF_Interrupt or PCS_INT (depending on PHY interface used).

If the software driver needs to reset GEM, it should set both the TX and RX Software Reset bits in the Software Reset Register, poll the register until both bits read low, and proceed to step C).

On Link Down the driver should disable the MAC and go to step F).

Note – The MAC should be disabled upon Link Down as soon as possible to minimize the chances of the link going up again without updating the MAC configuration accordingly.

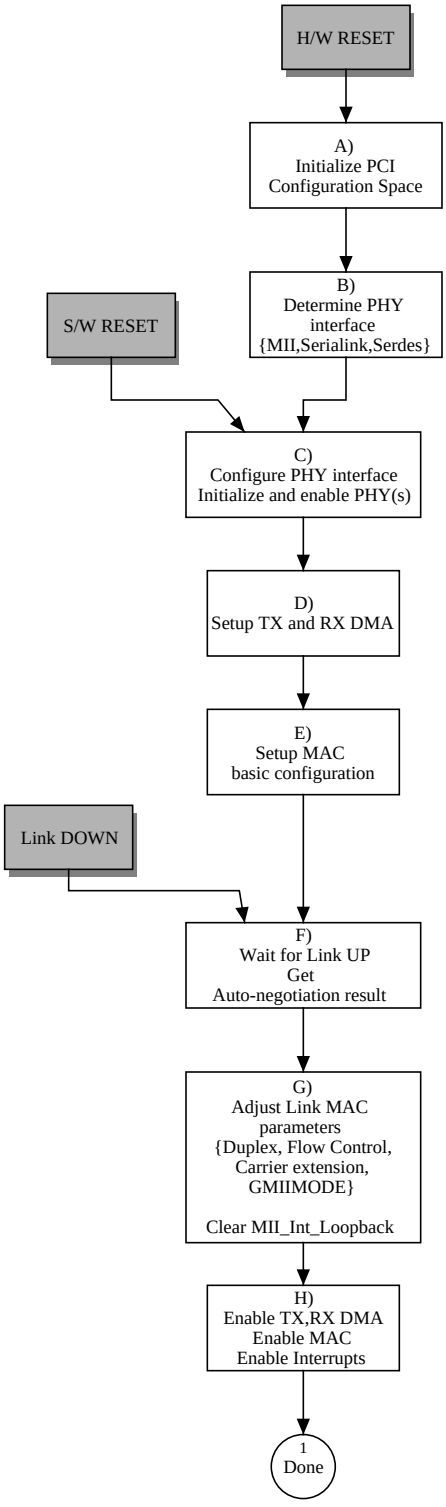


Figure 3-1 Initialization Flow

Table 3-34 MAC recommended initialization values

Address Offset	Initialization Value recommended	Actual Size (bits)	Register	Comment
MAC Registers				
0x6008	0x01BF0	17	Send Pause Command Register	PAUSE value large enough to get DMA moving again, small enough to block the link no worse than 16 retries on half duplex. Can use smaller values @10/100 Mbps.
0x6040	0x00	8	InterPacketGap0 Register	
0x6044	0x08	8	InterPacketGap1 Register	Per 802.3
0x6048	0x04	8	InterPacketGap2 Register	Per 802.3
0x604C	0x040	10	SlotTime Register	Per 802.3, change value later if needed for 802.3z carrier extension
0x6050	0x040	10	MinFrameSizeRegister	Per 802.3
0x6054	0x05EE	15	MaxFrameSize Register	Per 802.3
0x6058	0x07	10	PA Size Register	Per 802.3
0x605C	0x4	4	JamSize Register	Per 802.3
0x6060	0x10	8	Attempt Limit Register	Per 802.3
0x6064	0x8808	16	MAC Control Type Register	Per 802.3x
0x6080	Unique Station Address	16	MAC Address 0 Register	lower serial number bytes
0x6084		16	MAC Address 1 Register	lowest OUI byte, upper serial number byte
0x6088		16	MAC Address 2 Register	upper OUI bytes
0x608C	0x0000	16	MAC Address 3 Register	zero until used
0x6090	0x0000	16	MAC Address 4 Register	
0x6094	0x0000	16	MAC Address 5 Register	
0x6098	0x0180	16	MAC Address 6 Register	0x0001 per 802.3x
0x609C	0xC200	16	MAC Address 7 Register	0xC200 per 802.3x
0x60A0	0x0001	16	MAC Address 8 Register	0x0180 per 802.3x
0x60A4	0x0000	16	Address Filter 0 Register	zero until used
0x60A8	0x0000	16	Address Filter 1 Register	
0x60AC	0x0000	16	Address Filter 2 Register	
0x60B0	0x00	8	Address Filter 2&1 Mask Register	
0x60B4	0x0000	16	Address Filter 0 Mask Register	
0x60C0-0x60FC	0x0000	16	Hash Table 0 through 15 Registers	zero until used

Table 3-34 MAC recommended initialization values

Address Offset	Initialization Value recommended	Actual Size (bits)	Register	Comment
0x6100	0x0000	16	Normal Collision Counter	Clear all MAC counters
0x6104	0x0000	16	First Attempt Successful Collision Counter	
0x6108	0x0000	16	Excessive Collision Counter	
0x610C	0x0000	16	Late Collision Counter	
0x6110	0x0000	16	Defer Timer	
0x6118	0x0000	16	Receive Frame Counter	
0x611C	0x0000	16	Length Error Counter	
0x6120	0x0000	16	Alignment Error Counter	
0x6124	0x0000	16	FCS Error Counter	
0x6128	0x0000	16	RX code Violation Error Counter	

3.2.2 MAC Operational Modes

The following table indicates how the software driver selects the MAC layer network interface mode, in terms of Full Duplex vs. Half Duplex, and different loopback modes. The bits involved are:

- MII_Int_Loopback - XIF Configuration Register
- Disable_Echo - XIF Configuration Register
- Ignore_Carrier_Sense - TX MAC Configuration Register
- Ignore_Collisions - TX MAC Configuration Register
- No_Backoff - TX MAC Configuration Register
- Wrapback - PCS MII Control Register
- LOCKREF - Serialink Control Register
- LOOPBACK - Serialink Control Register

Table 3-35 Full Duplex and Loopback Configuration

	No_Backoff	Ignore_Collisions	Ignore_Carrier_Sense	Disable_Echo	MII_Int_Loopback	Wrapback, LOCKREF bits	LOOPBACK
Full Duplex	X	1	1	0		0	0
Half Duplex	0	0	0	Set to 1 if MII or GMII mode; Set to 0 if Serdes or Slink mode;	0	0	0
Internal Loopback at MII	X	0	0	0	1	0	0
Loopback at PCS	X	0	0	0	0	1	0
Loopback at Serialink	X	0	0	0	0	0	1

Note – The programming of the Duplex Mode can be a deliberate software driver decision, or might be based on the auto-negotiation outcome, as reported by the PCS layer.

Pinout and Mechanical 4

4.1 Functional I/Os Description

4.1.1 PCI Interface Signals

PCI_CLK

This clock provides the timing for all transactions on PCI bus and is an input to every PCI device. All PCI signals, except RST# and INTA# are sampled on the rising edge of this clock and all timing parameters are defined with respect to this edge. This clock runs up to 66 MHz.

RST#

This active low reset is used to bring PCI-specific registers, sequencers, and signals to a consistent state. Anytime RST# is asserted, all PCI output signals must be driven to their benign state. RST# may be asynchronous to CLK when asserted or de-asserted.

AD[31:0]

Multiplexed Address and Data. A bus transaction consists of an address phase followed by one or more data phases. The address phase is the clock cycle in which FRAME# is asserted.

AD[63:32]

Upper 32 bits of Address and Data bus. During the address phase, the upper 32-bits of the 64-bit address are transferred. Otherwise these bits are reserved. During the data phase, an additional 32-bits of data are transferred when REQ64# and ACK64# are both asserted.

C/BE[3:0]#

Bus Command and Byte Enables are multiplexed on the same PCI pins. During the address phase of a transaction C/BE[3:0]# define the bus command and during data phase they are used as byte enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. C/BE[0]# applies to byte 0 and C/BE[3]# applies to byte 3.

C/BE[7:4]#

Bus Command and Byte Enables used for 64 bit transfers. During the address phase, the actual command is transferred on C/BE[7:4]#. Otherwise, these bits are reserved. During a data phase, C/BE[7:4]# indicate which byte lanes carry meaningful data when REQ64# and ACK64# are both asserted. C/BE[4]# applies to byte 4 and C/BE[7]# applies to byte 7.

PAR

This signal provides even parity across AD[31:0] and C/BE[3:0]. PAR is stable and valid one clock after the address phase. For data phase, PAR is stable and valid one clock after either IRDY# is asserted on a write transaction or TRDY# is asserted on a read transaction. Once PAR is valid, it remains valid until one clock after the completion of the current data phase. The master drives PAR for address and write data phase; the target drives PAR for read data phase.

PAR64

This signal provides even parity across AD[63:32] and C/BE[7:4]#. PAR64 is valid one clock after the initial address phase when REQ64# is asserted and the DAC command is indicated on C/BE[3:0]#. PAR64 is valid one clock after the second address phase of a DAC command when REQ64# is asserted. PAR64 is stable and valid for data phases when both REQ64# and ACK64# are asserted and one clock after either IRDY# is asserted on a write transaction or TRDY# is asserted on a read transaction. Once PAR64 is valid, it remains valid for one clock after the completion of the data phase.The master drives PAR64 for address and write data phase; the target drives PAR64 for read data phase.

FRAME#

This signal is driven by the current master to indicate the beginning and the duration of an access. FRAME# is asserted to indicate that a bus transaction has started. While FRAME# is asserted, the data transfer continues. When FRAME# becomes de-asserted, the transaction is in the final data phase or has been completed.

IRDY#

The Initiator Ready signal indicates the bus master’s ability to complete the current data phase of transaction. IRDY# is used in conjunction with TRDY#. A data phase is completed on any clock both IRDY# and TRDY# are sampled as asserted. During a write cycle, IRDY# indicates that valid data is present on the AD[31:0] lines. During a read cycle, it indicates that the master is prepared to accept data. Wait cycles are inserted until both IRDY# and TRDY# are asserted together.

TRDY#

The Target Ready signal indicates the selected device's ability to complete the current data phase of transaction. TRDY# is used in conjunction with IRDY#. A data phase is completed on any clock both TRDY# and IRDY# are sampled asserted. During a read cycle, TRDY# indicates that valid data is present on the AD[31:0] lines. During a write cycle, it indicates that the target is prepared to accept data. Wait cycles are inserted until both IRDY# and TRDY# are asserted together.

STOP#

This signal indicates that the current target is requesting the master to stop the current transaction.

IDSEL

The Initialization Device Select signal is used as a "chip select" signal during configuration read and write transactions.

DEVSEL#

The Device Select signal, when actively driven, indicates the driving device has decoded its address as the target of the current access. As an input, DEVSEL# indicates whether any device on the bus has been selected.

REQ#

The Request signal is one of the signals monitored by the arbiter. It indicates that an agent (master) desires the use of the bus. This is a point-to-point signal. Every master has its own dedicated REQ#, which must be tri-stated while RST# is asserted.

REQ64#

This signal, when actively driven by a given bus master, indicates that this master desires to transfer data using 64-bit transactions. REQ64# has the same timing as FRAME#.

GNT#

The arbiter generates this signal, indicating to the master that access to the bus has been granted. This is a point-to-point signal. Every master has its own dedicated GNT#, which must be ignored while RST# is asserted.

ACK64#

Input to GEM. When actively driven by the device that has positively decoded its address as the target of the current access, indicates that the target is willing to transfer data using 64-bit transactions. ACK64# has the same timing as DEVSEL#.

PERR#

This signal is used for reporting data errors during all PCI transactions, except for a Special Cycle. The PERR# pin is sustained tri-state and must be driven active by the agent receiving data, two clocks following the data itself, if a data parity error is detected.

SERR#

The System Error output signal is used for reporting address errors, data parity errors on the Special Cycle command, or any other system error where the result will be “catastrophic”.

INTA#

The assertion and de-assertion of INTA# is asynchronous to the PCI_CLK. Used to requesting attention from the device driver. Once the INTA# signal is asserted, it remains asserted until the device driver clears the pending request.

M66EN

Input pin, when high it indicates that the bus segment operates at 66MHz. When low, the bus operates at 33MHz. The state of this pin is readable via GEM register space.

4.1.2 PCI PLL pins

PLL_BYPASS#

Input. The PCI clock PLL function is bypassed when low. PLL is engaged when this input is high. May be controlled using M66EN to select PLL for EPCI (66 MHz PCI).

PLL_CLK_OUT

Output.

PLL_EN

Input. The PCI clock PLL is reset when this signal is low. This input is usually tied to the PCI reset input (pin RST#).

IDDT

Input used for IDD test during device manufacturing. Tie low for normal operation.

PLL_VDD

Supply for PCI PLL.

PLL_VSS

VSS for PCI PLL.

4.1.3 GMII/SERDES Interface Signals

This group of signals is used for transceiver interface, using one out three possible interfaces: MII, GMII, and 10-bit SERDES.

CRS

Carrier Sense signal for MII/GMII. When asserted, the Carrier Sense signal indicates activity on the medium. This signal is asynchronous with respect to the MII/GMII clocks.

COL

Collision signal for MII/GMII. When asserted, the Collision signal indicates that a collision occurred on the medium. It remains asserted while the collision condition persist. This signal is asynchronous with respect to the MII/GMII clocks.

RXCLK

The Receive Clock input carries a continuous clock sourced by the transceiver, which provides the timing reference for sampling data, RXDV and RXER signals into GEM in MII/GMII modes. The frequency of this clock is the nibble rate for MII, and the byte rate for GMII (25MHz for 100 Mb/s over MII, 2.5MHz for 10 Mb/s over MII, and 125MHz for 1Gb/p over GMII).

RBC0

This input signal carries one phase of the SERDES receive clock, a 62.5MHz clock whose rising edges latch bytes 1 and 3 of the receive word. Bytes 0 and 2. are latched by a different phase of the receive clock RBC[1]. The receive data byte containing the Comma character (byte 0) is clocked by the rising edge of RBC[1]. RBC [0] and RBC [1] may be stretched during alignment.

RBC1

This input signal represents the second phase of the SERDES receive clock, a 62.5MHz clock whose rising edges latch bytes 0 and 2 of the receive word. RBC [0] and RBC [1] may be stretched during alignment.

RX[3:0]_S

Inputs carrying receive data lines 0 through 3 from the transceiver to the receive port of GEM. For MII/GMII these signals are synchronous to the receive clock, and data is transferred for every clock whenever Receive Data Valid is active.

For 10-bit SERDES modes, these lines are sampled alternately by RBC[0] and RBC[1] rising edges.

RX[7:4]_S

Inputs carrying receive data lines 4 through 7 from the transceiver to the receive port of GEM. For GMII these signals are synchronous to the receive clock, and data is transferred for every clock whenever Receive Data Valid is active. These signals are not used in MII mode.

For 10-bit SERDES modes, these lines are sampled alternately by RBC[0] and RBC[1] rising edges.

RX8_S

Input signal used as Received Data Valid in MII/GMII modes, synchronous to the receive clock. Driven by the transceiver to indicate that it is presenting recovered and decoded data on the receive data lines.

Used as receive data line eight (RX8) in 10-bit SERDES mode.

RX9_S

Input signal used as Received Error in MII/GMII modes. Asserted by the transceiver for one or more receive clock periods to indicate that an error was detected in the frame presently being transferred to GEM's receive port. This signal is synchronous to the receive clock.

Used as receive data line nine (RX9) in 10-bit SERDES mode.

TXCLK_MII

The Transmit Clock input carries a continuous clock sourced by the transceiver, which provides the timing reference for the transfer of the TXEN and transmit data signals from a GEM in MII/GMII modes. The frequency of this clock is the nibble rate for MII, and the byte rate for GMII (25MHz for 100 Mb/s over MII, 2.5MHz for 10 Mb/s over MII, and 125MHz for 1Gb/s over GMII).

TXCLK_S

This output carries the Transmit Clock as a buffered version of TXCLK_MII in MII/GMII modes.

In SERDES mode it carries the **TBC** (Transmit Byte Clock) output used to clock transmit data presented on TX[0:9]_S. This clock is derived from REFCLK, and has the same frequency and tolerance as REFCLK (typically 125MHz +/- 100 ppm).

TX[3:0]_S

Outputs carrying transmit data lines 0 through 3 from GEM to the transceiver. In MII/GMII modes for each transmit clock period in which TXEN ia asserted, data is transferred from GEM to the transceiver.

TX[7:4]_S

Outputs carrying transmit data lines 4 through 7 from GEM to the transceiver. In MII/GMII modes for each transmit clock period in which TXEN ia asserted, data is transferred from GEM to the transceiver.

TX8_S

In 10-bit SERDES modes this output represents TX8.

In MII/GMII modes this output carries the TXEN signal from GEM to the transceiver. It is driven by GEM to indicates to the transceiver that it is presenting data on the MII/GMII for transmission. This signal remains asserted while all nibbles/bytes to be transmitted are presented to the MII/GMII. This signal is synchronous to TXCLK_MII.

TX9_S

Output carrying TXER from GEM to the transceiver in MII/GMII modes.

In 10-bit SERDES modes this output represents TX9.

4.1.4 Transceiver Management Interface (MIF) Signals

These signals are used for either MII management and serial EEPROM interface. Up to two interfaces may be connected with their own bi-directional serial data line, while sharing the clock.

MDC

The Management Data Clock is sourced by the GEM MIF port to the XCVR as the timing reference for information on the MDIO0 and MDIO1 signals. This clock has a minimum high and low time of 160nS, and a minimum period of 400nS.

MDIO0

Management Data Input/Output 0 is a bi-directional signal between the transceiver and the GEM MIF interface for serial data transfer. This line carries the control information which is driven by the MIF interface with respect to MDC, and is sampled synchronously by the transceiver. Status information is driven by the transceiver synchronously with respect to the MDC clock, and is sampled synchronously by the MIF interface.

MDIO1

Management Data Input/Output 1 is a bi-directional signal between the transceiver and the GEM MIF interface for serial data transfer. This line carries the control information which is driven by the MIF interface with respect to MDC, and is sampled synchronously by the transceiver. Status information is driven by the transceiver synchronously with respect to the MDC clock, and is sampled synchronously by the MIF interface.

MII_BUF_EN

Output port used to enable MII interface external buffers if more than one interface per board is used. This pin is controlled by the MII_Buffer_OE bit in the *XIF Configuration Register.*

RSTOUT#

Reset output used to reset other board devices (PHYs). This pin is activated (low) whenever the RST# input is low or the RSTOUT bit in the *XIF Configuration Register* is set.

4.1.5 Dedicated SERDES Interface signals

The following signals have no other uses beyond SERDES interface.

EWRAP

Output used for controlling the physical layer loopback operation. When high the physical layer device is requested to internally loop the serialized transmit data to the de-serializer, and keep its transmit outputs static. Low for normal operation.

LCK_REF#

Output used to cause the SERDES device to lock the PLL to REFCLK.

COM_DET

Comma detect input. Indicates that data byte 0 of the first word contains a Comma character.

EN_CDET

Output. Used for enabling/disabling the byte alignment function of the external physical layer receiver. Enabled when high, disabled when low.

4.1.6 Signals used for SERDES and Serial Interface modes

REFCLK

Input clock. Nominal frequency is 125MHz and tolerance is +/- 100 ppm. REFCLK is used as the clock source for the Serialink block, and to derive TXCLK_S in SERDES mode. For designs that use only the MII interface, this clock may be tied low, and the Serialink block powered down.

SIG_DET

Signal detect input. Single-ended PECL input. Driven by the optical receiver. A logic “1” indicates normal optical input levels, a logic “0” indicates a low optical input level fault. Used by GEM to report “Remote Fault” to the other end of the link. This signal is used in SERDES and Serialink modes. Should be driven to a 3.3 V PECL logic “1” if the receiver does not support it (i.e. when using 8B/10B over copper).

VREFPECL

Voltage reference input for single-ended PECL level. Should be externally connected to a 2.0V +/- 5% reference. This reference is only used to define the SIG_DET input threshold.

4.1.7 Serial Interface Signals and Supplies

TXP, TXN

Differential PECL outputs carrying the serialized transmit data when EWRAP is low. When EWRAP is high, TXP is driven high and TXN is driven low. Must be externally terminated to a value between 50 to 75 ohms using parallel termination.

RXP, RXN

Differential PECL inputs. Serial receive data is selected to be received via RXP, RXN when EWRAP is low. Must be externally terminated to a value between 50 to 75 ohms using parallel termination.

IREF

Current bias reference for the Serialink analog bias generator.

TERMRES

Input connected to BGAVDD using an external resistor. The resistor value should be four times the termination impedance used for the serial transmit and receive lines.

VTXLO

Voltage reference used to determine the TXP/TXN minimum voltage swing.

TXBAVSS

VSS for Serialink transmit output section.

TXBAVDD

VDD for Serialink transmit output section.

RXBAVSS

VSS for Serialink receive input section.

RXBAVDD

VDD for Serialink receive input section.

TXAVSS

VSS for Serialink transmit analog section.

TXAVDD

VDD for Serialink transmit analog section.

RXAVSS

VSS for Serialink receive analog section.

RXAVDD

VDD for Serialink receive analog section.

BGAVSS

VSS for Serialink Bias Generator.

BGAVDD

VDD for Serialink Bias Generator.

GDAVSS

Serialink guard ring VSS

4.1.8 FCode PROM Interface Signals

P_A[15:0]

PROM address lines.

P_CS#

Chip Select line to the PROM.

P_D[7:0]

PROM data lines.

4.1.9 LEDs

LNKLED#

Link LED output. Active low.

FDPLXLED#

Full Duplex LED output. Active low.

TXLED#

Transmit activity LED output. Active low.

RXLED#

Receive activity LED output. Active low.

COLLED#

Collision LED output. Active low.

4.1.10 Testability

PO

IO ring testability output.

TN

Active low input. When high tristate outputs are disabled. Tie high for normal operation.

4.1.11 JTAG Signals

TCLK

TDI

TDO

TMS

TRST#

4.2 Pin Assignment

GEM pins fall into one of the following groups:

1. PCI Interface - 90 pins
2. PCI PLL - 6 pins
3. PCI VDD - 11 pins
4. GMII/SERDES interfaces - 36 pins
5. Serial Link/SERDES - 3 pins
6. Serial Link - 18 pins
7. JTAG - 5 pins
8. PROM Interface - 25 pins
9. LEDs - 5 pins
10. Test - 2 pins
11. Vdd - 12 pins
12. GND - 24 pins

Total = 237 pins

The “Pinout Table” shown below provides the physical pin assignment for the GEM ASIC. The TYPE column defines the drive characteristics. All I/Os use 3.3V supplies and signals that are 5V tolerant are identified as 5V compatible. The PCI VDD supplies are meant to be connected to the Vi/o supply of the PCI slot for universal PCI card design. The drive characteristics of PCI signals is TBD.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	PCI_CLK	K19	PCI clock	input 5V compat.
PCI	RST#	J17	PCI reset	input 5V compat.
PCI	AD0	B8	address / data	bi-directional 5V compat.
PCI	AD1	D9	address / data	bi-directional 5V compat.
PCI	AD2	C9	address / data	bi-directional 5V compat.
PCI	AD3	B9	address / data	bi-directional 5V compat.
PCI	AD4	D10	address / data	bi-directional 5V compat.
PCI	AD5	B10	address / data	bi-directional 5V compat.
PCI	AD6	A9	address / data	bi-directional 5V compat.
PCI	AD7	C11	address / data	bi-directional 5V compat.
PCI	AD8	A11	address / data	bi-directional 5V compat.
PCI	AD9	A12	address / data	bi-directional 5V compat.
PCI	AD10	B11	address / data	bi-directional 5V compat.
PCI	AD11	B14	address / data	bi-directional 5V compat.
PCI	AD12	C13	address / data	bi-directional 5V compat.
PCI	AD13	A13	address / data	bi-directional 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	AD14	C12	address / data	bi-directional 5V compat.
PCI	AD15	A14	address / data	bi-directional 5V compat.
PCI	AD16	D16	address / data	bi-directional 5V compat.
PCI	AD17	C17	address / data	bi-directional 5V compat.
PCI	AD18	B20	address / data	bi-directional 5V compat.
PCI	AD19	C19	address / data	bi-directional 5V compat.
PCI	AD20	D18	address / data	bi-directional 5V compat.
PCI	AD21	E17	address / data	bi-directional 5V compat.
PCI	AD22	C20	address / data	bi-directional 5V compat.
PCI	AD23	D19	address / data	bi-directional 5V compat.
PCI	AD24	F18	address / data	bi-directional 5V compat.
PCI	AD25	G17	address / data	bi-directional 5V compat.
PCI	AD26	E20	address / data	bi-directional 5V compat.
PCI	AD27	F19	address / data	bi-directional 5V compat.
PCI	AD28	G18	address / data	bi-directional 5V compat.
PCI	AD29	F20	address / data	bi-directional 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	AD30	G19	address / data	bi-directional 5V compat.
PCI	AD31	G20	address / data	bi-directional 5V compat.
PCI	AD32	K1	address / data	bi-directional 5V compat.
PCI	AD33	K3	address / data	bi-directional 5V compat.
PCI	AD34	K2	address / data	bi-directional 5V compat.
PCI	AD35	J1	address / data	bi-directional 5V compat.
PCI	AD36	J3	address / data	bi-directional 5V compat.
PCI	AD37	J4	address / data	bi-directional 5V compat.
PCI	AD38	H1	address / data	bi-directional 5V compat.
PCI	AD39	H2	address / data	bi-directional 5V compat.
PCI	AD40	H3	address / data	bi-directional 5V compat.
PCI	AD41	G1	address / data	bi-directional 5V compat.
PCI	AD42	G2	address / data	bi-directional 5V compat.
PCI	AD43	G3	address / data	bi-directional 5V compat.
PCI	AD44	F2	address / data	bi-directional 5V compat.
PCI	AD45	G4	address / data	bi-directional 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	AD46	F3	address / data	bi-directional 5V compat.
PCI	AD47	E1	address / data	bi-directional 5V compat.
PCI	AD48	E2	address / data	bi-directional 5V compat.
PCI	AD49	E3	address / data	bi-directional 5V compat.
PCI	AD50	D2	address / data	bi-directional 5V compat.
PCI	AD51	D3	address / data	bi-directional 5V compat.
PCI	AD52	B1	address / data	bi-directional 5V compat.
PCI	AD53	C3	address / data	bi-directional 5V compat.
PCI	AD54	B2	address / data	bi-directional 5V compat.
PCI	AD55	A2	address / data	bi-directional 5V compat.
PCI	AD56	B3	address / data	bi-directional 5V compat.
PCI	AD57	C4	address / data	bi-directional 5V compat.
PCI	AD58	D5	address / data	bi-directional 5V compat.
PCI	AD59	B4	address / data	bi-directional 5V compat.
PCI	AD60	C5	address / data	bi-directional 5V compat.
PCI	AD61	D7	address / data	bi-directional 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	AD62	B4	address / data	bi-directional 5V compat.
PCI	AD63	C6	address / data	bi-directional 5V compat.
PCI	C / BE[3:0]	E20,A18,B13,A10 (msb to lsb)	command / byte enable for lower 32 bits	bi-directional 5V compat.
PCI	C / BE[7:4]	A8,A6,C7,B6 (msb to lsb)	command / byte enable for upper 32 bits	bi-directional 5V compat.
PCI	PAR	A16	parity for lower 32 bits	bi-directional 5V compat.
PCI	PAR64	A5	parity for upper 32 bits	bi-directional 5V compat.
PCI	FRAME#	C15	cycle frame	bi-directional 5V compat.
PCI	IRDY#	A17	initiator ready	bi-directional 5V compat.
PCI	TRDY#	B16	terminator ready	bi-directional 5V compat.
PCI	STOP#	C14	stop	bi-directional 5V compat.
PCI	IDSEL	D20	initialization device select	input 5V compat.
PCI	DEVSEL#	C16	device select	bi-directional 5V compat.
PCI	REQ#	H19	request	output 5V compat.
PCI	REQ64#	C8	request 64 bit transfer	output 5V compat.
PCI	GNT#	H20	grant	input 5V compat.
PCI	PERR#	A15	parity error	bi-directional 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PCI	SERR#	B15	system error	output 5V compat.
PCI	INTA#	J18	interrupt A	open drain output 5V compat.
PCI	ACK64#	B7	acknowledge for 64 bit transfer	input 5V compat.
PCI	M66EN	B12	66MHz Enable line	input 5V compat.
PCI PLL	PLL_BYPASS#	J19	PCI clock PLL bypass	input
PCI PLL	PLL_CLK_OUT	L1		output
PCI PLL	PLL_EN	K18	PLL enable/reset	input Schmitt Trigger
PCI PLL	IDDT	T17	IDD test	input
PCI PLL	PLL_VDD	K17		supply
PCI PLL	PLL_VSS	J20		supply
GMII/SERDES	CRS	V6	MII: CRS (Carrier) GMII: CRS (Carrier) SERDES: GND	input 5V compat.
GMII/SERDES	COL	U7	MII: COL (Collision) GMII: COL (Collision) SERDES: GND.	input 5V compat.
GMII/SERDES	RXCLK	V7	MII: RXCLK (receive clock) GMII: RXCLK (receive clock) SERDES: GND	input 5V compat.
GMII/SERDES	RBC0	W6	MII: GND GMII: GND SERDES:RBC0 (receive byte clock phase1)	input 5V compat.

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
GMII / SERDES	RBC1	Y6	MII: GND GMII: GND SERDES:RBC1 (receive byte clock phase2)	input 5V compat.
GMII / SERDES	RX[3:0]_S	W8,V8,Y7,W7 (msb to lsb)	MII: RXD[3:0] receive data GMII: RXD[3:0] receive data SERDES: RX[3:0]	input 5V compat.
GMII / SERDES	RX[7:4]_S	W9,V9,U9,Y8 (msb to lsb)	MII: GND GMII: RXD[7:4] receive data SERDES: RX[7:4]	input 5V compat.
GMII / SERDES	RX8_S	Y9	GMII: RXDV (receive data valid) GMII: RXDV (receive data valid) SERDES: RX8	input 5V compat.
GMII / SERDES	RX9_S	W10	MII: RXER (receive error) GMII: RXER (receive error) SERDES: RX9	input 5V compat.
GMII / SERDES	TXCLK_MII	W13	MII: TXCLK (transmit clock) GMII: TXCLK (transmit clock) SERDES: GND	input 5V compat.
GMII / SERDES	TX8_S	V12	MII: TXEN (transmit enable) GMII: TXEN (transmit enable) SERDES: TX8	output
GMII / SERDES	TX9_S	U12	MII: TXER (transmit error) GMII: TXER (transmit error) SERDES: TX9	output
GMII / SERDES	TX[3:0]_S	W11,Y11,Y10,V10 (msb to lsb)	MII: TXD[3:0] transmit data GMII: TXD[3:0] transmit data SERDES: TX[3:0]	output
GMII / SERDES	TX[7:4]_S	W12,Y12,U11, V11 (msb to lsb)	MII: GND GMII: TXD[7:4] transmit data SERDES: TX[7:4]	output
GMII / SERDES	TXCLK_S	Y13	MII: GND GMII: TXCLK_OUT SERDES: TBC (Transmit byte clock)	output 5V compat.
GMII / SERDES	MDC	Y5	MII: MDC (management clock) GMII: optional SERDES: GND	output

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
GMII / SERDES	MDIO0	Y3	MII: MDIO0 (management data) GMII: optional SERDES: GND	bi-directional 5V compat.
GMII / SERDES	MDIO1	W5	MII: MDIO1 (management data) GMII: optional SERDES: GND	bi-directional 5V compat.
GMII / SERDES	MII_BUF_EN	V4	MII Buffer Enable	output
GMII / SERDES	RSTOUT#	L2	Reset Output	output
SERDES	EWRAP	W14	Wrap control output	output
SERDES	COM_DET	V13	Comma character detection indication	input 5V compat.
SERDES	EN_CDET	V14	Comma character detection enable	output
SERDES	LCK_REF#	Y15	Lock to Reference	output
Serial / SERDES	REFCLK	Y14	Reference clock	input 5V compat.
Serial / SERDES	SIG_DET	W15	Signal detection indication from optical receiver	single-ended PECL input
Serial / SERDES	VREFPECL	Y16	Reference voltage for single-ended PECL level. Externally connected to 2.0 V.	single-ended PECL input
Serial	TXP, TXN	M20, M19	Serial transmit data	PECL differential output
Serial	RXP, RXN	N20, N19	Serial receive data	PECL differential input
Serial	IREF	R19	Current Bias Reference	input
Serial	TERMRES	P17	Termination Resistor Reference	input

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
Serial	VTXLO	R20	Low-swing Voltage Reference	input
Serial	TXBAVSS	L19	VSS for transmit output	supply
Serial	TXBAVDD	M18	VDD for transmit output	supply
Serial	RXBAVSS	N18	VSS for receive input	supply
Serial	RXBAVDD	M17	VDD for receive input	supply
Serial	TXAVSS	L20	VSS for transmit analog section	supply
Serial	TXAVDD	L18	VDD for transmit analog section	supply
Serial	RXAVSS	P19	VSS for receive analog section	supply
Serial	RXAVDD	P20	VDD for receive analog section	supply
Serial	BGAVSS	R18	VSS for bias generator	supply
Serial	BGAVDD	P18	VDD for bias generator	supply
Serial	GDAVSS	K20	Guard Ring VSS	supply
PROM	P_A0	P4	PROM address line	output
PROM	P_A1	T1	PROM address line	output
PROM	P_A2	R2	PROM address line	output
PROM	P_A3	P3	PROM address line	output
PROM	P_A4	R1	PROM address line	output
PROM	P_A5	P2	PROM address line	output
PROM	P_A6	P1	PROM address line	output
PROM	P_A7	N3	PROM address line	output
PROM	P_A8	N2	PROM address line	output
PROM	P_A9	N1	PROM address line	output
PROM	P_A10	M4	PROM address line	output

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
PROM	P_A11	M3	PROM address line	output
PROM	P_A12	M2	PROM address line	output
PROM	P_A13	M1	PROM address line	output
PROM	P_A14	L4	PROM address line	output
PROM	P_A15	L3	PROM address line	output
PROM	P_D0	W1	PROM data bus	input 5V compat.
PROM	P_D1	V2	PROM data bus	input 5V compat.
PROM	P_D2	U3	PROM data bus	input 5V compat.
PROM	P_D3	V1	PROM data bus	input 5V compat.
PROM	P_D4	U2	PROM data bus	input 5V compat.
PROM	P_D5	T3	PROM data bus	input 5V compat.
PROM	P_D6	U1	PROM data bus	input 5V compat.
PROM	P_D7	T2	PROM data bus	input 5V compat.
PROM	P_CS#	R3	PROM chip select	output
LEDs	LNKLED#	U14	Link LED	open drain
LEDs	FDPLXLED#	V15	Full Duplex LED	open drain
LEDs	TXLED#	W16	Transmit activity LED	open drain
LEDs	RXLED#	Y17	Receive activity LED	open drain
LEDs	COLLED#	V16	Collision LED	open drain

Table 4-1 Pinout Table

INTERFACE	PIN NAME	BALL #	FUNCTION	TYPE
Test	PO	W17	I/O ring testability	output
Test	TN	V17	Tristate output disable, active low	input
JTAG	TCLK	V3	JTAG clock	input
JTAG	TDI	U5	JTAG data in	input with built-in pullup
JTAG	TDO	W4	JTAG data out	output
JTAG	TMS	W2	JTAG test mode select	input with built-in pullup
JTAG	TRST#	Y2	JTAG reset	input Schmitt-trigger with pullup
Supplies	PCIVDD	H18, E19, C18, D14, D12, C10, A7, A4, C2, F1, J2	PCI buffers VDD from I/O pins (can be 5V or 3.3V)	
Supplies	VDD	D6,D11,D15,F4, F17,K4,L17,R4, R17,U6,U10,U15	VDD	
Supplies	VSS	A1,D4,D8,D13, D17,H4,H17,N4, N17,U4,U8,U13, U17,J9,J10,J11,J12 ,K9,K10,K11,K12, L9,L10,L11,L12, M9,M10,M11, M12	VSS	

4.3 Electrical Specifications

4.3.1 Absolute Maximum ratings

Table 4-2 Absolute maximum ratings

Symbol	Parameter	Limit	Unit
VDD	DC Power Supply Voltage	-0.3 to 3.9	V
Vin, Vout	DC Input, Output Voltage	-1.0 to VDD + 0.3	V
Vin	DC Input Voltage for 5V compatible inputs	-1.0 to 6.5	V
Tstg	Storage Temperature	-40 to 125	DegC
Pd	Power dissipation	3.0	Watts

Note – Stresses beyond those listed in the above table may cause physical damage to the device and should be avoided.

4.3.2 Recommended Operating Conditions

Table 4-3 Recommended Operating Conditions

Symbol	Parameter	Limit	Unit
VDD	DC Power Supply Voltage	3.3 +/- 5%	V
Vin, Vout	DC Input, Output Voltage	0 to VDD	V
Tj	Junction Temperature	0 to 105	DegC
Ta	Operating Ambient Temperature	0 to 70	DegC

4.3.3 DC Characteristics

Table 4-4 DC Specification for PCI Interface

Symbol	Parameter	Conditions	Min	Max	Unit
Vil	Input Low Voltage		0.5*VDD	VDD+0.5	V
Vih	Input High Voltage		-0.5	0.3*VDD	V
Iil	Input Leakage Current Low	0<Vin<Vss	-10	10	uA
Vol	Output Low Voltage	Iol=1500 uA		0.1*VDD	V
Voh	Output High Voltage	Ioh=-500 uA	0.9*VDD		V
Cout	Output Pin Capacitance	excludes package		10	pF

Table 4-5 DC Specifications of PCS/GMII, EPROM, JTAG, LED, MDIO Interfaces

Symbol	Parameter	Conditions	Min	Max	Unit
Vil	Input Low Voltage		Vss-0.5	0.8	V
Vih	Input High Voltage		2.0	6.5	V
Iil	Input Leakage Current	Vin = 0 to VDD	- 10	10	uA

Symbol	Parameter	Conditions	Min	Max	Unit
Vt	Schmitt-trigger, Input Switching Threshold Voltage			2.0	V
Vt+	Schmitt-trigger, Input Switching Threshold Voltage, rising edge			2.0	V
Vt-	Schmitt-trigger, Input Switching Threshold Voltage, falling edge			1.0	V
Cin	Input Capacitance	excludes package		3.0	pF
Vol	Output Low Voltage		0	0.4	V
Voh	Output High Voltage		2.4	VDD	V
Ioh	Output High Current	bt6f@VOL=0.4V		13.25	mA
Iol	Output Low Current	bt6f@VOL=0.4V	4	9.6	mA
Ioz	High Z Leakage current (open drain)	Vpad=5.5V	-10	10	uA
Cout	Output Capacitance	excludes package		3	pF

Table 4-6 DC Specification for PECL Interfaces

Symbol	Parameter	Conditions	Min	Max	Unit
Vstx	TXN,TXP Peak-to-peak Differential Output Swing		0	1600	mV
Vsrx	RXN,RXP Peak-to-peak Differential Input Sensitivity		400	1600	mV
Vih	Input High Voltage for single ended PECL	Va - Vref = 50 mV		Vref + 50	mV
Vil	Input Low Voltage for single ended PECL	Va - Vref = 50 mV	Vref - 50		mV
Idd	Current in VDD for single ended PECL	iddtn = 0	1.4	0	mA
Iih	Input leakage current LOW for single ended PECL	Vin = VDD	-10		uA
Iil	Input leakage current HIGH for single ended PECL	Vin = VSS		10	uA

4.3.4 Environmental ELectrical Protection

Table 4-7 Environmental Electrical Protection

ESD	Latch UP
Minimum	Minimum
2KV	150 mA

4.3.5 AC Characteristics

4.3.5.1 PCI Interface

Table 4-8 PCI Clock AC Timing Specifications

Symbol	Parameter	66 MHz Min	66 MHz Max	33 MHz Min	33 MHz Max	Unit
T_cyc	pci_clk Cycle Time	15	30	30	DC	ns
T_high	pci_clk High Time	6		11		ns
T_low	pci_clk Low Time	6		11		ns

Table 4-9 PCI Input AC Timing Characteristics

Symbol	Parameter	66 MHz Min	66 MHz Max	33 MHz Min	33 MHz Max	Unit
T_su	Input Setup Time to bused inputs	3		7		ns
T_su(ptp)	Input Setup Time to pci_req_l, pci_gnt_l	5		10,12		ns
T_hold	Input Hold Time from pci_clk	0		0		ns
T_rst	Reset Active Time after Power Stable	1		1		ms
T_rst(clk)	Reset Active after pci_clk stable	100		100		us
T_rst(off)	Reset Active to Output Float Delay		40		40	ns

Input signals are PCI_CLK, RST#, IDSEL, GNT#. These signals except RST# are synchronous to the positive edge of PCI_CLK. RST# assertion and de-assertion is asynchronous toPCI_CLK.

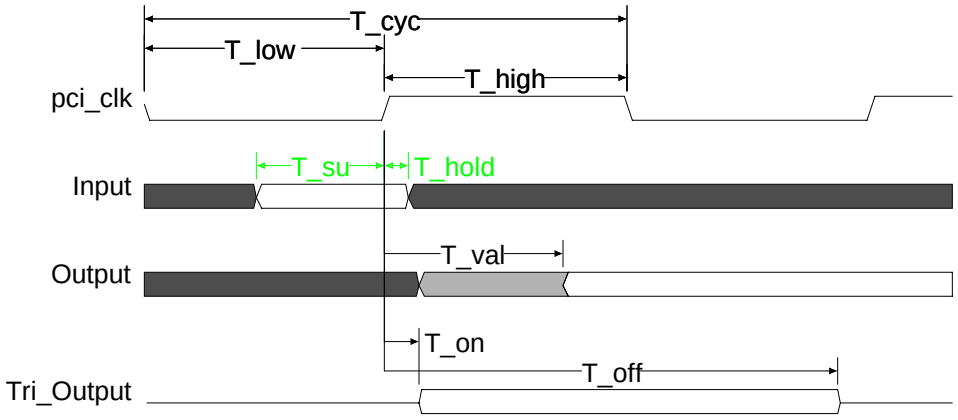
Table 4-10 PCI Output AC Timing Characteristics

Symbol	Parameter	66 MHz Min	66 MHz Max	33 MHz Min	33 MHz Max	Unit
T_val	pci_clk to Signal Valid Delay - bused	2	6	2	11	ns
T_val(ptp)	pci_clk to Signal Valid Delay - pci_req_l, pci_gnt_l	2	6	2	12	ns
T_on	Float to Active Delay	2		2		ns
T_off	Active to Float Delay		14		28	ns
T_rrsu	REQ# to RST# setup time	10*T_cyc		10*T_cyc		us
T_rrh	REQ# to RST# hold time	0	50	0	50	ns

PCI output signals are REQ#, INTA#. The assertion and de-assertion of INTA# is asynchronous to PCI_CLK.

PCI bi-directional signals are AD[63:0], C/BE[7:0], LOCK#, PAR, FRAME#, IRDY#, PERR#, STOP#, PAR64#, REQ64#, ACK64#. These signals must meet the timing requirements mentioned in the PCI Input and PCI Output Timing Characteristics tables.

Figure 4-1 PCI Timing Waveform



4.3.5.2 PCS Interface

Table 4-11 PCS Receive Bus AC Specification

Symbol	Parameter	125 MHz Min	125 MHz Typ	125 MHz Max	Unit
Tfreq	RX_CLK Frequency		62.5		MHz
Tdrift	RX_CLK Drift Rate	0.2			us/MHz
Tsetup	RX_S data bus, COM_DET, LOCKREFN setup to RX_CLK	2.5			ns
Thold	RX_S data bus, COM_DET, LOCKREFN hold from RXCLK	1.5			ns
Tduty	RXCLK duty cycle	40		60	%
Ta-b	RXCLK Skew	7.5		8.5	ns
Tdo	EN_CDET setup to RXCLK	2.0			ns

Figure 4-2 PCS Receive Interface Timing Waveforms

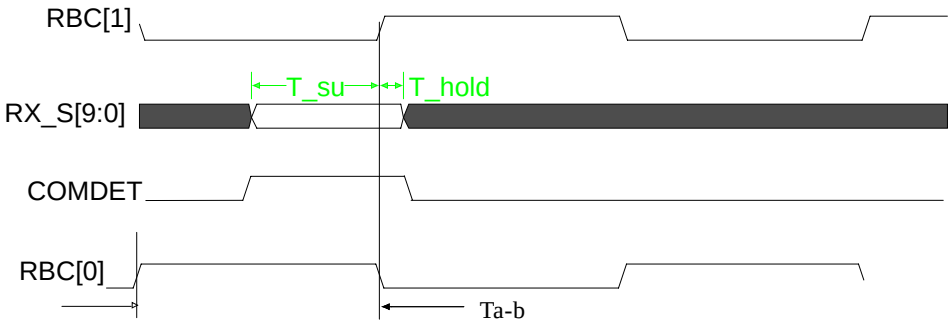


Table 4-12 PCS Transmit Bus AC Specification

Symbol	Parameter	125 MHz Min	125 MHz Typ	125 MHz Max	Unit
Tperiod_ref	REFCLK Period, +/- 100 ppm tolerance		8		ns
Tperiod	TXCLK_S Period, +/- 100 ppm tolerance		8		ns
Tsetup	TX_S data bus setup to TXCLK_S	2.0			ns
Thold	TX_S data bus hold from TXCLK_S	1.0			ns
Tduty	TXCLK_S duty cycle	40		60	%

4.3.5.3 GMII Interface

Table 4-13 GMII General AC Specifications

Symbol	Parameter	Conditions	Min	Max	Units
Tr	Clock rise time	Vil_ac(max) to Vih_ac(min)		1.00	ns
Tsetup	Clock fall time	Vil_ac(min) to Vih_ac(max)		1.00	ns
-	Clock Slew Rate (rising)	Vil_ac(max) to Vih_ac(min)	0.6		V/ns
-	Clock Slew Rate (falling)	Vil_ac(min) to Vih_ac(maxn)	-0.6		V/ns
Vil_ac	Input Low Voltage AC			0.70	V
Vih_ac	Input High Voltage		1.90		V

Table 4-14 GMII AC Specifications for Transmit signals

Symbol	Parameter	Min	Max	Units
Tperiod	TXCLK_S Period	7.50		ns
Thigh	TXCLK_S Time High	2.50		ns
Tlow	TXCLK_S Time Low	2.50		ns
Tsetup	TXD, TX_EN, TX_ER Setup to rising edge of TXCLK_S	2.50		ns
Thold	TXD, TX_EN, TX_ER Hold fromrising edge of TXCLK_S	0.50		ns

Table 4-15 GMII AC Specifications or Receive signals

Symbol	Parameter	Min	Max	Units
Tperiod	RXCLK Period	7.50		ns
Thigh	RXCLK Time High	2.50		ns

Tlow	RXCLK Time Low	2.50		ns
Tsetup	RXD, RX_DV, RX_ER Setup to RXCLK	2.00		ns
Thold	RXD, RX_DV, RX_ER Hold to RXCLK	0.00		ns

Note – CRS and COL signals are asynchronous in MII and GMII Interfaces.

Table 4-16 MDIO/MDC AC Specification

Symbol	Parameter	Min	Max	Units
Tperiod	MDC Period	400		ns
Thigh	MDC Time High	160		ns
Tlow	MDC Time Low	160		ns
Tsetup (driven by GEM)	MDIO to MDC, when MDIO is being driven by GEM	10		ns
Thold (driven by GEM)	MDIO from MDC, when MDIO is being driven by GEM	10		ns
Tsetup (driven by PHY)	MDIO to MDC, when MDIO is being driven by PHY	100		ns
Thold (driven by PHY)	MDIO from MDC, when MDIO is being driven by PHY	0		ns

Table 4-17 EPROM AC Specification

Symbol	Parameter	Min	Max	Units
Tsetup	P_A bus Setup to PCICLK	7.00		ns
Thold	P_A bus Hold to PCICLK	2.00		ns
Tdo	PCICLK to P_D data bus valid	1.0	7.0	ns

Table 4-18 JTAG AC Specification

Symbol	Parameter	Min	Max	Units
Tclk	TCLK	2	10	MHz
Tsetup	TDI, TMS setup to Jtag TCLK	5		ns
Thold	TDI, TMS hold to Jtag TCLK	10		ns
Tdo	TDO output delay from Jtag TCLK	TCLK/2 + 7.49	TCLK/2 + 15.90	ns
Trst_	TRST_ assertion time	5 TClk		

4.4 Mechanical Information

4.4.1 Package info & drawings

GEM's package is a four layer 272 pin PBGA. Figure 4-3 is a top view of the package solder mask.

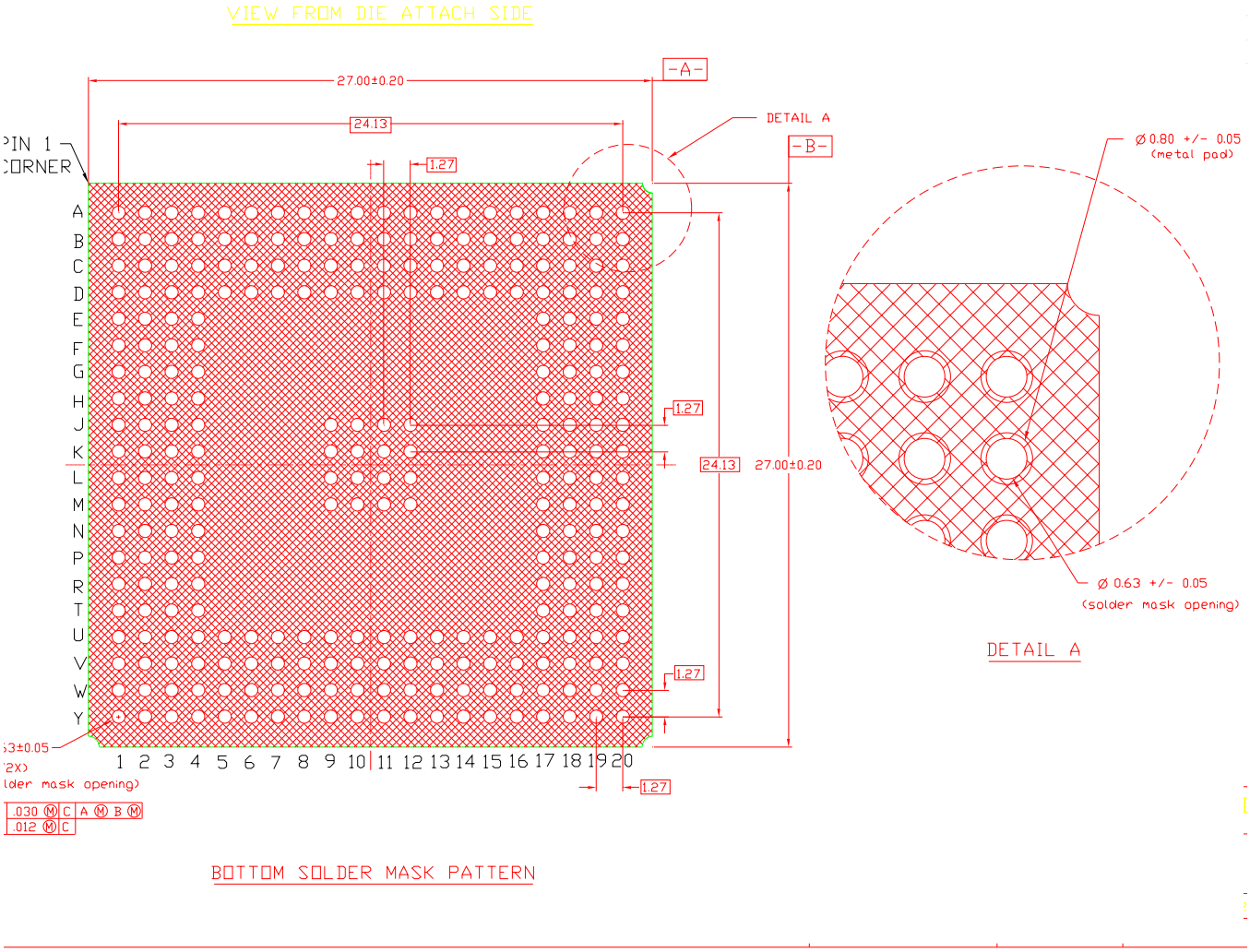


Figure 4-3 Pacakge footprint - Preliminary

4.4.2 Package characteristics

Table 4-19 Pin and Size Characteristics

Package type	Pin Count	Cavity Size	Signals	Die Range
VG 272 PBGA+	272	0.430x0.430 in.	231	6.89-9.86 mm

Table 4-20 Thermal Resistance

no heat sink	0 lfpm	200 lfpm	400 lfpm	600 lfpm
Theta Ja (Degrees C / Watt)	16.4	13.9	13.1	12.3

with external heat sink	0 lfpm	200 lfpm	400 lfpm	600 lfpm
Theta Ja (Degrees C / Watt)	12.3	8.5	6.9	6.1

Theta Jc	2.4

Table 4-21 Pin Loading

	Min	Max	Unit
Resistance	69	117	mO
Inductance	2.7	8.2	nH
Capacitance	0.96	1.54	pF
Crosstalk Capacitance	0.22	0.34	pF